AFRL-IF-RS-TR-2005-376
Final Technical Report
November 2005

HIGH LEVEL FUSION IN THE CYBER DOMAIN

University at Buffalo

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

STINFO FINAL REPORT

AFRL-IF-RS-TR-2005-376 has been reviewed and is approved for publication

APPROVED: /s/

DOUGLAS M. BOULWARE
Project Engineer

FOR THE DIRECTOR: /s/

JOSEPH CAMERA, Chief
Information & Intelligence Exploitation Division
Information Directorate

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 074-0188*

1. AGENCY USE ONLY (Leave blank)	**2. REPORT DATE** NOVEMBER 2005	**3. REPORT TYPE AND DATES COVERED** Final May 05 – Aug 05

4. TITLE AND SUBTITLE HIGH LEVEL FUSION IN THE CYBER DOMAIN	**5. FUNDING NUMBERS** C - FA8750-05-1-0207 PE - 62702F PR - 558B TA - II WU - RS
6. AUTHOR(S) Moises Sudit, Shanchieh Yang, Michael Kuhl, Adam Stotz, Michael Holender, Jared Holsopple, Eric Bohannon, and Jason Kistner	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) University at Buffalo Center for Multisource Information Fusion Buffalo New York 14260	**8. PERFORMING ORGANIZATION REPORT NUMBER** N/A
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Research Laboratory/IFEA 525 Brooks Road Rome New York 13441-4505	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER** AFRL-IF-RS-TR-2005-376

11. SUPPLEMENTARY NOTES
AFRL Project Engineer: Douglas M. Boulware/IFEA/(315) 330-4599/ Douglas.Boulware@rl.af.mil
Summer Research Fellowship Final report.

12a. DISTRIBUTION / AVAILABILITY STATEMENT APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.	**12b. DISTRIBUTION CODE**

13. ABSTRACT ***(Maximum 200 Words)***

The objective of this report is to summarize the work performed during the summer at AFRL, Rome Research site by a team of researchers from SUNY at Buffalo and Rochester Institute of Technology (RIT). The team began their research with lessons learned from the testing of INFERD v1, and applied research methods to overcome the deficiencies found. These deficiencies and an overview of the old methodology of INFERD are discussed along with the results of this research. Work discussed in the report includes attack identification and tracking techniques, identification of cyber threats and modeling and generation of cyber environments for generation of data sets.

14. SUBJECT TERMS Cyber Indications and Warning, Cyber Situation Awareness, Threat, Intentions, Fusion	**15. NUMBER OF PAGES** 57
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	**18. SECURITY CLASSIFICATION OF THIS PAGE** UNCLASSIFIED	**19. SECURITY CLASSIFICATION OF ABSTRACT** UNCLASSIFIED	**20. LIMITATION OF ABSTRACT** UL

Table of Contents

List of Figures

List of Tables

1. Introduction and Motivation

1.1 INFERD Mission

The Information Fusion Engine for Real-time Decision Making (INFERD)'s objective is to provide real-time situational assessment of a domain of interest to facilitate decision making of an analyst by detecting the current state and future threat of situations within the environment through a hierarchical fusion of sensory data.

1.2 Summer Objective

The objective of this summer's work was to take lessons learned from the testing of INFERD v1, and apply research methods to overcome the deficiencies found. These deficiencies and an overview of the old methodology of INFERD are discussed in the Section 1.3. The results of this research and a description of the new methodology are given in the Section 2. At the conclusion of the summer program an event was held at AFRL called "Hackfest 2005", in which our team was lucky enough to have an opportunity to test the new system developed over the course of the summer. Our team's experiences taken away from this initial test of the system will be given in the Section 2.1. INFERD, both its old and current version, was designed and developed in such a way as to make it portable across multiple domains. For the sake of this report, and the program it was developed under, its application will be in the context of the cyber domain in which it is a part of a larger system called the Event Correlation for Cyber Attack Recognition System (ECCARS). Figure 1 summarizes the different major research areas and which members of the team worked on each of the areas,

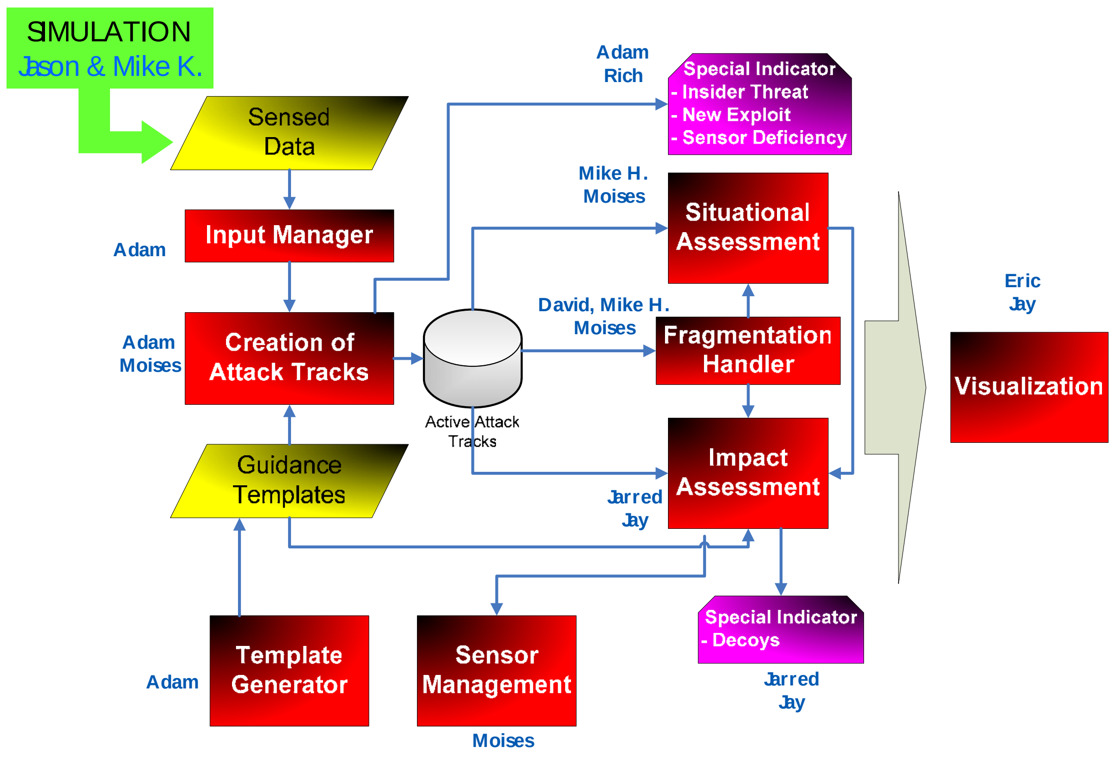


Figure 1: Work Breakdown Structure

Team

- Dr. Moises Sudit (Team Manager)
  Managing Director
  Center for Multisource Information Fusion – University at Buffalo
  Expertise: Optimization, Graph Theory and Heuristics
- Dr. Shanchieh Yang
  Assistant Professor
  Computer Engineering- Rochester Institute of Technology
  Expertise: Information Theory, Visualization and Sensors
- Dr. Michael Kuhl
  Associate Professor
  Industrial and Systems Engineering – Rochester Institute of Technology
  Expertise: Simulation and Optimization
- Ph.D. Students
  - Adam Stotz (Industrial Engineering, University at Buffalo)
  - Michael Holender (Industrial Engineering, University at Buffalo)
- Bachelor and Masters Students
  - Jared Holsopple (Computer Engineering, Rochester Institute of Technology)
  - Eric Bohannon (Computer Engineering, Rochester Institute of Technology)
  - Jason Kistner (Industrial Engineering, Rochester Institute of Technology)
- David Sudit (Computer Engineering, Northwestern University)

1.3 Introduction of INFERD v1

The main idea of INFERD v1 in the ECCARS system is to fuse a stream of incoming heterogeneous IDS alerts to a database of Attack Templates and rank them according to how likely they are taking place. These Attack Templates are developed a priori and represent the possible complex multistage attacks which may take place within the environment. This ranking of Attack Templates is presented to the user in a method which gives them a situational awareness of their network and aids them in their decision process of how to mitigate the threat. The decoupled architecture along with the standardized XML interfaces allows for facilitated plug-and-play integration with external applications and allows us to leverage research being performed by other teams such as a graph matching application developed by 21st Century Technologies.

Attack Templates, shown in Figure 2 define the cyber attacks monitored in the alert stream. These structures are created a priori in a hierarchical top-down fashion manually by a subject matter expert (SME). These attack graphs are specified in and loaded from an XML file when the system is started. It is implausible to expect a human operator to generate all possible cyber attacks manually which makes the system dependent upon the success of a related future research effort to automatically generate the exhaustive set of possible attacks for a given network.

In v1 these templates are very rigid in the sense that specific IP addresses and alert information must be encoded into the template set making the number of templates exponential with respect to the number of machines in a given network.

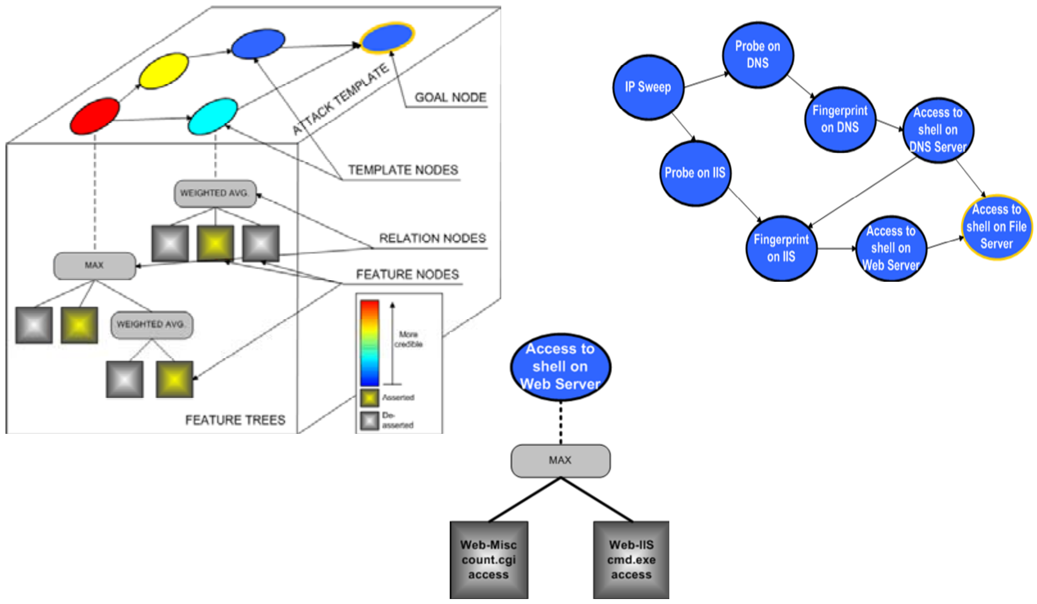


Figure 2: Attack Templates

Fusion is performed on the templates themselves in a bottom-up fashion as shown in Figure 3. L0 fusion is defined to be the correlation of alerts to *Feature Nodes* in which the results are the assertion of the Feature Nodes which appropriately characterize the alert which was fused. Once a Feature Node is asserted, L1 fusion takes over and calculates a *credibility* value for the *Template Node* containing the *Feature Tree* which contained the newly asserted Feature Node. After the L1 credibility value is calculated for the Template Node, any Attack Template containing that node then performs an L2 fusion process which calculates an overall credibility for the Attack Template. These Attack Template credibilities are then used as a mechanism on which to rank the complete a-priori set of Attack Templates. Focus for the rest of this section will now shift to the problems inherent in this aforementioned methodology as discovered in the blind test held at CUBRC last May.

The major deficiency found during the blind test, and it was not a surprise, was the necessity for automatic template generation. When not knowing the scenario that will be played out, an exhaustive set of possibilities must be generated to ensure the successful capturing of that scenario and thus results in an unmanageable number of a-priori templates for a large network.

The other major deficiency found was not being able to distinguish between attacking parties. Since the templates only characterized *what* was happening and not *who* was doing it, a major request of the customer was not able to be met.

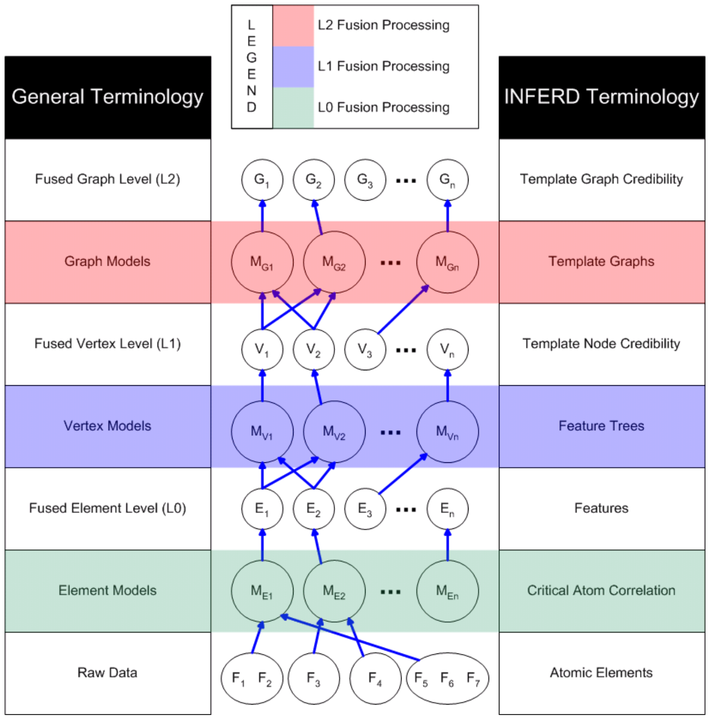


Figure 3: Multi-tier Fusion Framework

2 Summer work and creation of INFERD v2

INFERD v2 is the result of the summer research focusing on the mitigation of these problems within the INFERD framework. The most significant change made, requiring an almost complete rewrite of the existing INFERD code, was instead of ranking an exhaustive set of a-priori templates, we allow an a-priori attack model (or small set of models), which we call a Guidance Template, guide the dynamic instantiation and evolution of *Attack Tracks*. Each of these Attack Tracks is associated with a single attacking party and is an accumulation of events triggered by their actions. In this new methodology the old notion of Attack Templates like the one shown in Figure 2, can now be considered an Attack Track whose instantiation is of the Guidance Template shown in Figure 4.

Guidance Templates consist of nodes and arcs, where nodes consist of a Feature Tree as in the old methodology and represent attack steps in the aggregate Multi-stage attack model. Arcs represent feasible transitions between attack steps and help define for Attack Tracks whether new sensor information coming into the system should be fused to which Attack Tracks. Both the nodes and arcs are attributed with Template Variables which take on values when those components are instantiated within an Attack Track. Through this dynamic process of instantiation, we can now track attacking parties throughout their attack evolution and we no longer need an exhaustive set of attack step / target machine combinations.

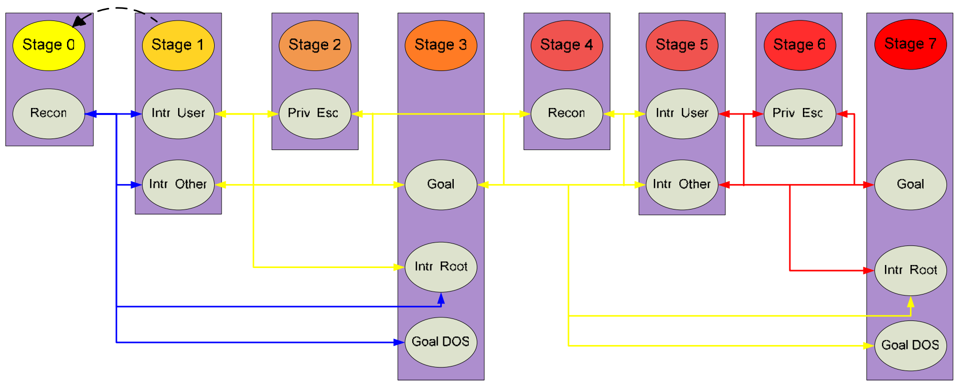


Figure 4: Multistage Cyber Attack Model

Attack Tracks from a traditional Information Fusion perspective are hypotheses about the situation of the environment and the aggregation of the evaluation of these hypotheses can be thought of the situational assessment (L2 fusion). Since we are now performing fusion on dynamic attack tracks instead of a-priori templates, new SA algorithms were also researched this summer and are implemented in a modular plug-and-play fashion within the new INFERD framework. This is necessary because there is a low probability that two different environments will require exactly the same situational assessment, even if their models follow a common schema.

In the new methodology it is very possible for a single alert to be fused to multiple Attack Tracks and in the cyber context this is a problem. Since Attack Tracks represent the steps of a single attacking party, how can a single event detected by an IDS sensor be triggered by multiple attackers? To overcome this problem, the new version of INFERD introduces a notion of Hyper Attack Tracks (HATs).

As shown in Figure 5 when a single event is fused to multiple attack tracks, that event then becomes a Hyper Node and a Hyper Attack Track is created whose root is the Hyper Node. This introduces a problem of fragmentation because in reality it was a single party who executed the attack, even though it was fused to multiple attacking parties during INFERD's processing.

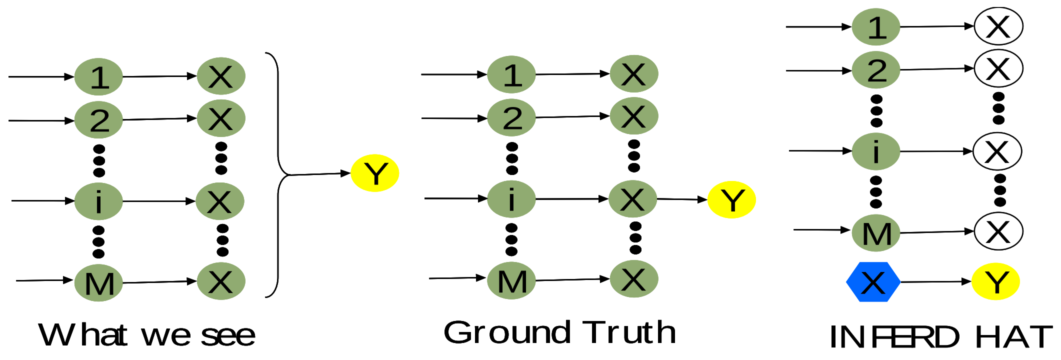


Figure 5: Hyper Attack Tracks

Just as plug-and-play algorithms for SA were researched over the summer, de-fragmentation algorithms were also researched. Because it is likely that no single metric will perform best in all situations, the de-fragmentation process built into INFERD v2 leverages from concepts introduced by Multiple Managed Algorithms (MMAs). Here, the plug-and-play algorithms can be loaded and executed on Hyper Attack Tracks and any one decision will not be finalized until the combined estimations from all managed algorithms cross some configurable threshold of confidence.

2.1 Hackfest 2005

Hackfest this past year, consisted of four teams each containing a red and blue group. The red groups of each team were responsible for breaking into the other teams' networks and modifying a special file called a "flag", while the blue teams were responsible for defending their own flags. This is a very attack rich environment which can provide a good stress test to the installed systems.

Since ground truth is not yet known for the events of Hackfest, the accuracy of INFERD v2 cannot yet be determined. The new system did, however, perform well from a computational efficiency perspective. Over the 13 hour period of activity approximately 1.5 million alerts were produced. This alert volume dwarfs that of even current large scale networks and INFERD kept up with the pace being able to process approximately 500 alerts per second.

3 <u>Level 2 Fusion Measurements</u>

3.1 Measurements at the Feature Tree Level (Level 1 Fusion)

When data is input to INFERD, it is linked to Feature Nodes based on the type of alert that is sensed. In this particular domain an alert occurs or it does not, hence the given values are binary. Feature Nodes can have an "importance weight (u)" assigned to them. Upon completion of the aforementioned process, INFERD obtains an aggregated value of the Feature Nodes (w) for the Template Node at the top of the Feature Tree which then becomes a node on an Attack Track. The Credibility Factor (CF) decays with time according to Equation 1. This time decay is calculated based on the entire Attack Track. For instance, once a new node is attached to a current Attack Track, its time is "restarted" (t = 0) and the decay commences from there.

$$CF_{ij}^{k} = w^{k} * \frac{1-e^{-\left(L_i - t \middle/ \alpha_i\right)}}{1-e^{-L_i / \alpha_i}} \tag{1}$$

For Template Node k of Attack Track i of Guidance Template j

w^k = aggregated value of the Feature Nodes

L_i = lifetime of Attack Track i

t = elapsed time since the most recent attack on Attack Track i

α^i = decay rate of Attack Track i

The lifetime and decay rate of each Attack Track is chosen by the user. Figure 6 shows the effect of different decay rates using the example Feature Tree in Figure 3 assuming a lifetime $L^i = 20$.

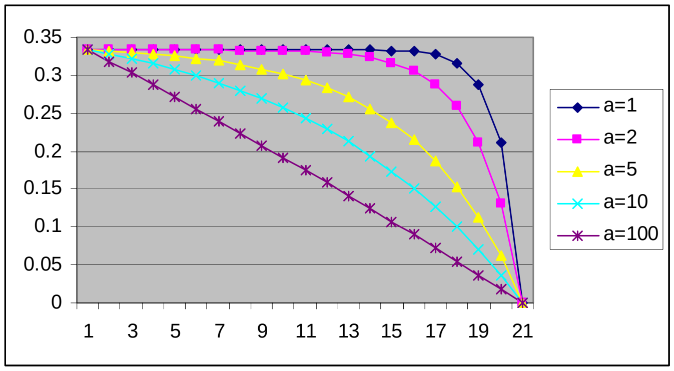


Figure 6: Effects of decay rate – as α increases, decay becomes more linear

3.2 Measurements of the Attack Tracks (Level 2 Fusion)

The Level 2 measures use the time-decayed Credibility Factors obtained above to help the user assess the current situation of his/her network. The following defines the mathematics used in calculating the L2 measures:

$GT^j = (V^j, E^j) \qquad for\ \ j = 1,\ldots,n$

where

GT^j=Guiding Template graph j ∀j

V^j=Vertices of Guiding Template graph ∀j

E^j=Edges of Guiding Template graph ∀j

$G_{it}^j = (V_{it}^j, E_{it}^j) \quad for\ \ i = 1,\ldots,m_{jt}$

where

m_{jt} = Number of Instances of Guiding Template j active at time t ∀j,t

$G_{it}^j \subseteq GT^j$ is a subset of a Guiding Template with

$S = \bigcup_{h=1}^{H} S_h$ = Set of stages

$S_h \bigcap S_f = \varnothing \ \forall h, f \in H \ \ (h \neq f)$

$v \in \bigcup_{h=1}^{H} S_h \ \ \forall v \in \bigcup_{j=1}^{n} V^j$

r_h=value for each stage h ∈ H

$s(v) = \{r_h : v \in S_h\} \ \ \forall v \in \bigcup_{j=1}^{n} V^j$

$h_{ijt}^{\max} = \underset{k \in V_{it}^j}{Max}\{h \in H\}$

3.2.1 Depth

The Depth of an attack measurement gives an indication of how close your adversary is to his/her possible target. It helps answer the question; how far into the attack is this particular hacker? Equation 2 describes the calculation.

$$D_{it}^{j} = (r_{h_{ijt}^{\max}} + \{CF_k^i(t,\alpha_k) : v_k \in S_{h_{ijt}^{\max}}\})/(H+1) \quad \forall i, j, t \tag{2}$$

3.2.2 Breadth

The Breadth of an attack measures how much of the entire possible scope of the attack has already taken place or how "full" the attack is in comparison to the entire Guidance Template. The denominator of Equation 3 ($B_t^{\max}$) calculates the total possible Credibility assuming all nodes of the Guidance Template are instantiated, it is a normalization factor. The numerator sums up the Credibilities of the Attack Track.

$$\begin{aligned} B_t^{\max} &= \sum_{h=1}^{h_{ijt}^{\max}} \underset{\substack{i=1,\ldots,m_{jt} \\ j=1,\ldots,n}}{Max} \{|S_h \cap V_{it}^j|\} && \forall t \\ B_{ijt} &= \frac{\sum_{v_k \in V_{it}^j} CF_{ki}^j(t,\alpha_k)}{B_t^{\max}} && \forall i, j, t \end{aligned} \tag{3}$$

It can be argued that a hacker is more knowledgeable if the depth is large and the breadth is small implying that they have taken a direct route toward their goal. The converse may be true as well; if the breadth is large and the depth is small, one could infer the hacker may not be as advanced. This notion is still up for debate, but if true can be used for behavioral modeling to help answer questions on Impact Assessment (Level 3 fusion) or the Fragmentation problem as introduced in Section 2 and discussed in detail in Section 4.

3.2.3 Reliability

Reliability is an entropy-based function that calculates how sure we are that this particular attack is continuing to take place. Tsallis [17, 18] formulates a general form of entropy (Equation 4) containing a parameter q that can be optimized for any specific domain. For most Optimization and Information Theoretical problems, q → 1 thus giving Shannon's [15] Entropy function (Equation 5).

$$H_q = k\frac{1-\sum_{i=1}^{N} p_i^q}{q-1} \qquad \sum_{i=1}^{N} p_i = 1, k > 0 \tag{4}$$

$$H = -k\sum_i p_i \log p_i \tag{5}$$

Assume we have K equi-probable states such that H = log K, we want the overall number of states represented by a decreasing function of the variable $x = \sum_{k=1}^{|N|} CF_{ki}^{j}$. We obtain the following extremes using the above method for calculating entropy (H):

$$\begin{aligned} K &= |N| - x + 1 = |N| - \sum_{k=1}^{|N|} CF_{ij}^{k} + 1 \\ H_{\min} &= H(x = |N|) = \log K \,|_{CF_{ij}^{k}=1 \forall k} = \log(1) = 0 \\ H_{\max} &= H(x = 0) = \log K \,|_{CF_{ij}^{k}=0 \forall k} = \log(|N|+1) \end{aligned} \tag{6}$$

Since a single value of entropy has little interpretive value, we borrow the notion of Relative Entropy from Pierce and Shannon [13] to obtain our Reliability measure as seen in equation 7.

$$E = \frac{H_{\max} - H(x)}{H_{\max} - H_{\min}} = \frac{\log(|N|+1) - \log(|N| - \sum_{k=1}^{|N|} CF_{ij}^{k} + 1)}{\log(|N|+1)} \tag{7}$$

3.3 Example of the Situational Assessment Measures

Assume we have the following Guidance Template (Figure 7) with four Attack Tracks instantiated upon it.

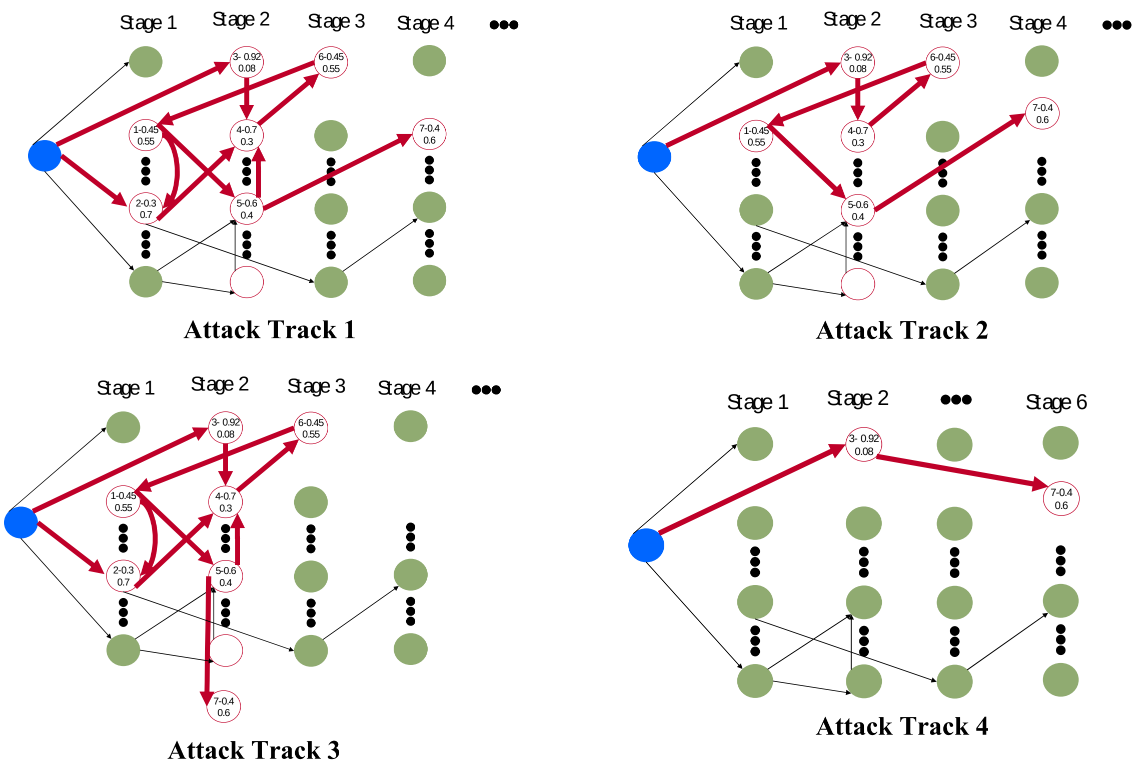


Figure 7: There are four active Attack Tracks upon the same Guidance Template

Using Equation 2 we can calculate the Depth of Attack Track 1 (upper left). Assuming that there are a total of eight stages in the given Guidance Template and the furthest point of the attack is at stage 4, half way through toward the goal, then the depth would be 0.489.

Using Equation 3 we calculate the Breadth of Attack Track 1. We show all Attack Tracks in Figure 7 because the Breadth calculation is reliant on each of the active Attack Tracks. B = 0.424. This also makes sense since this Attack Track appears to be rather "full" in comparison to the rest.

Finally we calculate the Reliability of Attack Track 1 using Equation 7. R = 0.312; this suggests that there is about a 31% chance that this particular attack is continuing to take place, thus it may not be a major concern in respect to others.

4 Handling Fragmentation

The purpose of the Fragmentation Handler is to be able to assist security personnel determine who is attacking a computer network in a multi-stage attack. Hackers can use other's computer identities to attack a different computer and this makes it hard for security to correctly determine who the original attacker is (generally referred to as a stepping stone). Through research and discussion with Subject Matter Experts (SME's) we have found that hackers can be characterized using indicators to determine their behavior. We hope to use this characterization to assist analysts using this system to make the decision rather than providing them with one.

4.1 Fragmentation

Fragmentation is when two or more attack tracks end up attacking the same computer. Then from that computer another attack is detected. The question is "Who is responsible for the attacks?" We use this fragmentation handler to give the user some numbers to maybe help them determine where the attacker is located. Figure 8 shows an example of the fragmentation problem.

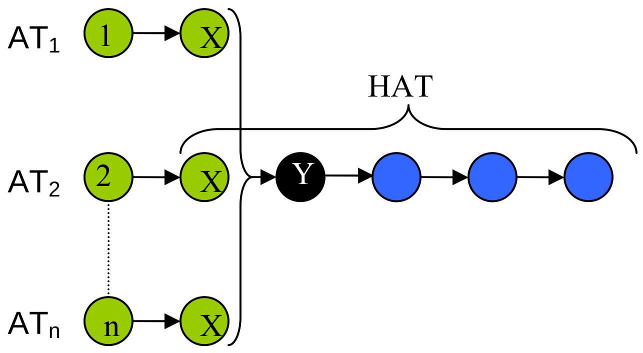


Figure 8: Example of Fragmentation Problem

Let's say you have one to n attackers. We denote each attack track as AT_1 to AT_n, where n must be at least two. All of the attackers attack computer X. In turn one of them uses the identity of computer X to attack computer Y. The problem now is that when you see the attack at computer Y you have n possibilities of who is continuing their attack to Y. We will call X the Hypernode. The track after and including X will be called HAT or Hypernode Attack Track. Our definition of a Hypernode is a Template node in two or more active AT's from which the Guidance Template has correlated a new attack. That is, it is a node representing a subset of AT's that cannot be differentiated or are ambiguous as attacks progressing from that point forward. There are some interesting cases that happen when we implement Hypernodes. First, if a Hypernode does occur then we only need to add one more attack track. So the above example would turn into Figure 9. Secondly, if another Hypernode occurs after X then we again only need to add one more AT to the end with the starting node being the Hypernode. These rules will help keep the number of AT's we have to a minimum.

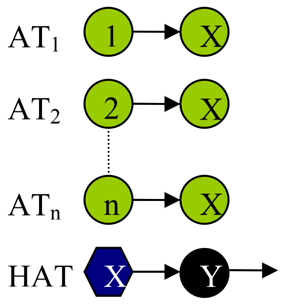


Figure 9: Example of Analysis

To decipher which of the n AT's is continuing his or her attack we will use three different methods. First we will do deletion or discarding of attack tracks. The second step will be the merging of the attack tracks in the set that are left over. Lastly we will take the remaining attack tracks and rank them using indicators. We believe these indicators can be used to compare the characteristics of the AT's to the HAT and tell the user which AT we believe might have continued the attack to Y.

4.2 Deletion/Discarding

There are situations where we have found that certain attack tracks can be discarded. By removing these attack tracks first it will help keep calculation time as quick as possible.

4.2.1 Temporal Discarding

The first discarding method is temporal discarding. Temporal discarding happens when the time an attack track happens is after that when the HAT already exists. For example, let's look at Figure 8 again. Let's say the attack to X in attack track two (AT_2) happened one hour ago. The attack from X to Y in the HAT was seen three hours ago.

We can say that since the HAT attack already happened before attacker two even reached point X that AT_2 can be discarded out of the possibilities.

4.2.2 Decay Discarding

Our second deletion method is decay discarding. Decay discarding deals with time as did temporal discarding. The difference between the two is that decay discarding looks at how long ago the attack to the Hypernode occurred. For example, if AT_2 happened 4 months ago and we set our threshold to be 3 months. Then at this moment the attack from X to Y occurs. We assume that since the AT_2 happened such a long time ago and the attack on Y just happened that it is very likely that it is not attacker two continuing the attack. Other methods are being looked into as we feel that eliminating attack tracks before the computation stage will help us give more accurate results and in a quicker time.

4.3 Merging

The Deletion/Discarding rules are absolute and will work consistently, Merging and Indicators, have no guarantees as to them working right every time. These are just methods that will aide the user in deciding who is more likely to have continued the attack past the Hypernode. Three merging techniques are Temporal, Signature and Commonality merging.

4.3.1 Temporal Merging

Temporal merging occurs when one and only one of the attack tracks has the most recent attack time and the other AT's in the set have happened a considerable time ago. For example, in Figure 10 we see there are three ATs and the HAT, which has just occurred. If AT_1 and AT_3 happened one month ago and AT_2 occurred one day ago then we are claiming that AT_2 is more probable to be the culprit to have continued the attack.

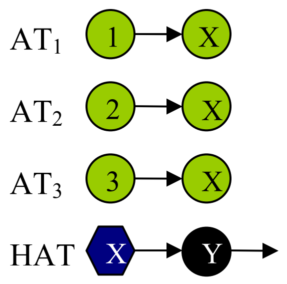


Figure 10: Example of Temporal Merging

4.3.2 *Signature Merging*

The second type of merging is Signature. Signature relates to the personality of an attacker. The system looks at what the attacker has done up to the point of the Hypernode. Then it looks at what has been going on after the Hypernode. Those two personalities are then compared to each other and we claim that if one is much more similar than the others that it will most likely be the one continuing the attack.

4.3.3 *Commonality Merging*

Commonality merging is the third type of merging we have discovered. Figure 11 shows an example where at some point in AT_2 there is a node called W. From that node W the attacker had attacked computer Y. The attacker of AT_2 still went ahead and attacked X at some point in time. Two situations arise if this occurs. First if the attack from W to Y happened before the attack from X to Y then we claim that AT_2 most likely was not the one continuing the attack because he or she already had access to Y and did not need to go through X to get access to Y. The second situation is if the W to Y attack happened after the X to Y attack then we claim it is likely AT_2. We say this because if AT_2 went from X to Y he or she would already know how to infiltrate the system and could do so easily from a different computer like W.

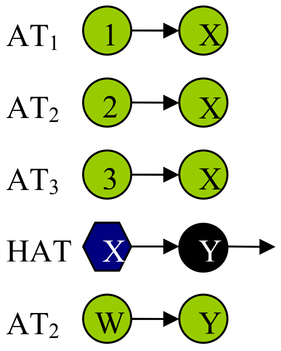


Figure 11: Example of Signature Merging

4.4 *Indicators*

After using the deletion and merging techniques to lower the number of attack tracks to a minimum we start to use indicators to give us probabilities of who is the attacker. After many discussions with subject matter experts we came across three indicators which they believe can be used to characterize traits of hackers. The three indicators are: Operating system usages and patterns, inter-event times between attacks and signatures.

4.4.1 *Operating systems*

Every attack registered by the sensors is connected with a specific Operating system (OS). A non-experienced hacker might only know how to use certain operating systems and might use the same one through out the attack. An experienced hacker would know how to use many operating systems and to try to avoid being tracked down would jump around operating systems. These two statements helped us come to find two measurements, total percentage and randomness, which we feel show the user who is the more likely attacker.

Total percentage (TP) is a measurement used to see how much of the same OS is being used in each AT to the HAT. It is measured using Equation 8.

$$P_{HAT_s} = {|N_s|}\big/{|N|}, N_s \subseteq N \qquad TP_j = \frac{\sum_{s \in S} (1 - |P_{HAT_s} - P_{AT_s^j}|)}{|S|}, \forall j \qquad (8)$$
$$P_{AT_s^j} = {|N_s^j|}\big/{|N^j|}, N_s^j \subseteq N$$

where N is the number of nodes in the HAT not including the Hypernode, N_s is the number of nodes that contain OS type s, N^j is the number if nodes in ATj, N_s^j is the number of nodes that contain OS type s in ATj and S is just the number of different OS types. We exclude the Hypernode because after testing we saw that it artificially increased the results.

OS type	**HAT**	**AT$_1$**	**AT$_2$**	**AT$_3$**	**1-(HAT-AT$_1$)**	**1-(HAT-AT$_2$)**	**1-(HAT-AT$_3$)**
A	3/5	3/5	1/5	2/7	1	0.6	0.69
B	2/5	1/5	0	1/7	0.8	0.6	0.74
C	0	1/5		1/7	0.8		0.86
D	0		4/5			0.2	
E	0			1/7			0.86
F	0						
G	0			1/7			0.86
H	0						
I	0						
J	0			1/7			0.86
Avg.					0.867	0.467	0.812

Table 1: Example Matches

Here is an example to explain how the math shows us who is attacking. Let's say that we find there are ten OS types, S = {A,B,C,…,J}. The HAT and AT's look as follows: HAT:**A**-A-A-A-B-B, AT$_1$:A-A-A-B-C-**A**, AT$_2$:D-D-D-D-A-**A**, AT$_3$:E-G-J-A-B-C-A-**A**. The bold letter 'A' represents the Hypernode. The value for A in the HAT is 3/5 because we have a total of five nodes not including the Hypernode and we have three of OS type A. Then you take that value and subtract it from the total percent of A in AT$_1$ and then take the difference to get the similarity. Next we take an average to get the average similarity over all the OS types. We continue this process and find that AT$_1$ is 86.7 percent similar to the HAT and AT$_3$ is very similar as well at 81.2 percent.

This measurement will show us a lot about the hackers but we came across a situation that caused this measurement to fail. Total percent failed when the OS in both attack tracks were completely different from that of the HAT. For example, if we use the same set of OS types as above and the tracks look like, HAT: E-F-G-H-I, AT_1: A-A-A-A-E and AT_2: A-B-C-D-E, then both values come out to be zero similarity. By looking at this situation we found that AT_2 would be more likely to have continued his attack as AT_1 seems to just know how to use OS A. So we came up with a randomness measurement using entropy. The entropy (E) measurement is done by taking the entropy of each AT and subtracting it from the entropy of the HAT. We divide that value by the max entropy with is –log(1/S), where S is the number of OS types. This keeps the value normalized. After normalizing and subtracting the value from one to get the similarity of the randomness of each AT compared to the HAT.

$$H = -\sum p_s \log p_s \tag{9}$$

So in the example we gave above AT_1 would end up having very little similarity in this value and AT_2 would have a substantial amount of similarity, which is the desired result.

4.4.2 Inter-event Time

Inter-event time is the time between each attack. The attackers sometimes have patterns of how much time it takes them to do each attack and this could be seen as a continuation into the HAT which could point us to the correct attacker. The way we have decided to use inter-event times is to form regressions and calculate the error. For each AT a regression will be formed. It can be a line, exponential or power regression. We decide what kind of regression it will be by looking at the correlation coefficients until we either find one that is ninety-five percent correlated or we take the kind that is the most correlated out of the three. Then the HAT inter-event times are plotted onto the regressions. Whichever regression produces the least amount of error is the AT we associate with the HAT. To find these regressions we just use the normal regression formulas as shown on equation 10 and 11.

$$m = \frac{n\sum(xy) - \sum x \sum y}{n\sum x^2 - (\sum x)^2} \ldots b = \frac{\sum y - m(\sum x)}{n} \quad (10)$$

$$R^2 = \left(\frac{n\sum(xy) - \sum x \sum y}{\sqrt{n\sum x^2 - (\sum x)^2} * \sqrt{n\sum y^2 - (\sum y)^2}} \right)^2 \quad (11)$$

where m is the slope of the line, b is the intercept and R^2 is the correlation coefficient. The x values are just the node number in each attack track and the y values are the inter-event times. Here is an example of how we use regression. Let's say that the attack tracks and inter-event times are shown in Figure12.

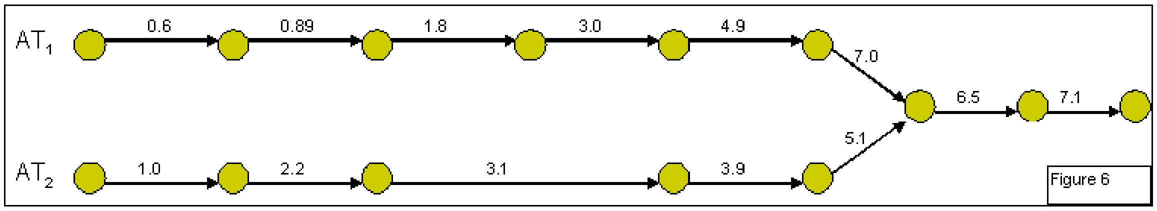


Figure 12: Example Attack Track/Inter-Event Times

Given these times we find that AT_1 is 99.2 percent correlated with the exponential regression and that AT_2 is 99.5 correlated with the linear regression. Next we calculate what the predicted inter-event times should be for the HAT. Take the averages of the error found between the actual and predicted inter-event time for both the attack tracks. In this case AT_1 has an average error of 10.605 and AT2 has an average of 0.275. To get the similarity value we evaluate the following:

Similarity value for AT_1 = 1 – (10.605/(10.605 + 0.275)) =0 .025

Similarly for AT_2 we replace the numerator with 0.275 and get 0.975. So we find that AT_2 is 97.5 percent correlated to the HAT and AT_1 is only 2.5 percent correlated and we say that AT_2 is probably the hacker that continued the attack.

4.4.3 Signatures

We believe that signatures will be the most important part in fragmentation. We think that the concept of Conceptual Spaces could be used to give the user a very accurate comparison of traits of hackers. We think that if you set the HAT to be the object and then take each AT and compare that to the HAT using concept spaces that the value given will be the most accurate way to help the user decide between hackers.

5 <u>Impact and Threat Assessment</u>

Since many commercial tools focus primarily on assessing the threat of individual alerts and alerts occurring on the same machine, a new approach is needed to assess the impact and the threat of incoming attacks spanning multiple machines. The follow sections discuss how cyber attacks fit into level 3 of JDL data fusion model, the applicability of stochastic models, and a proposed novel algorithm for multi-stage cyber attack threat assessment.

5.1 Application to Level 3 Fusion

In the JDL model, level 3 fusion is characterized by the following three attributes -- capability, opportunity, and intent [2]. The combination of these three attributes can lead to estimates of what the attacker plans to do and what the impact of such an action would be.

The following analyzes each of the three attributes of level 3 fusion, and discusses how they could be applied to cyber attacks.

5.1.1 Capability

Capability is what the attacker is able to do. The capability of an attacker depends on its educational level and physical resources. There are many different levels of hackers. They range from normal computer users who simply download hacking scripts to a very advanced hacker who has extensive knowledge of vulnerabilities in networking and operating systems. An advanced hacker will be able to perform more complex attacks than a hacker who is simply running a script. Therefore, the type of attack could indicate the level of the hacker. If a complex attack is detected, the hacker is likely to be more advanced and thus a higher threat level could be associated with that attack. However, if a common script attack occurs, the attacker could be a novice hacker running a script – it could still, however, be an advanced hacker. Therefore, the complexity of the attack can only establish a lower bound on the educational level of the hacker.

A hacker must have access to a computer capable of connecting to the network to be hacked. In most cases, the computer is one connected to the internet; however it may be possible that the hacker has access to a workstation in the internal network.

5.1.2 Opportunity

Opportunity is what the attacker can do next. Opportunity can be determined by the vulnerabilities of the system being attacked and what information the attacker currently has access to. In cyber attacks, opportunity applies to what the hacker is able to do next. Consider a case where a hacker knows how to exploit a service on a server to obtain administrative rights. While the hacker has the *capability* of exploiting that service, he cannot exploit that service until he knows that it is running on the server. If he determines that the service is running on the server, then he has the *opportunity* to exploit that service. Now, if the hacker is unaware of how to exploit a particular service, then the opportunity to exploit the service still exists, but the likelihood of it occurring is small because of the limited capability.

5.1.3 Intent

Intent is what the attacker plans to do and is very difficult to identify in the cyber attack domain. Some hackers are known as ethical hackers, where they hack into a network without any malicious intentions. However, other hackers hack into a system to steal, delete, or modify critical data. Unfortunately, the intentions of the hackers cannot be identified until a malicious act has occurred. Once a hacker gains the privileges he needs to accomplish his goal, to our knowledge there is no model that predicts whether the hacker will modify critical files or just simply leave the system.

If a hacker has already performed a malicious act, it is probably safe to assume that the hacker has malicious intentions. However, if the hacker has not done any malicious attacks, it is almost impossible to determine the intentions of the hacker.

5.2 Stochastic Modeling

The use of stochastic models such as Bayesian networks and hidden Markov models were motivated in [14]. While [14], focused on terrorist attacks, this idea could be extended to cyber attacks. This section will discuss how stochastic models have been or could be developed for cyber attacks. Potential drawbacks of using such models are also discussed.

5.2.1 Bayesian Networks

Bayesian networks are a technique to model uncertainty, and thus could be used to predict the future if the past evidence is indicative of what could happen in the future. The states of such a network have a certain probability of occurring. Bayesian networks are characterized by the fact that the probability of being in one possible state of the network is dependent upon the previous states. The probability of being in one state, X, depends on the set of previous states, $\mathbf{Y}$, i.e. $P(X|\mathbf{Y})$. For a more rigorous definition and mathematical formulation of Bayesian Networks, please see [6].

Bayesian networks can be applied to cyber attacks by defining a course of action in which the hacker can penetrate the network. This course of action could be defined by the types of attacks the hacker could perform or the information the hacker has compromised during his attack. Phillips and Swiler suggest in [12], that this course of action can be generated based on the topology of the network, the services running on each machine, configuration, user groups, attacker profile and other network characteristics. Each edge would be assigned a probability of transition, so the most likely path (or the *n* most likely paths) could be computed using well-known algorithms. This can be used to estimate the opportunity and intent. They also mention that different attacker profiles can be created to estimate the capability.

While this is a very comprehensive idea, it may not be realistic for enterprise networks. Even if the course of action was generated for a complex network, the assignment of probabilities is not a trivial task. The probabilities could be assigned by subject matter experts, but due to complexity, this could be very time consuming and inaccurate. The probabilities could also be trained, but cyber attack data is very limited. With the set of all possible cyber attacks constantly increasing, which makes the course of action non-stationary, this training data may not be useful to predict the future.

The authors admit that this idea may not be scalable if the levels of aggregation are not chosen properly (i.e. machine vs. subnet level). However, they do suggest ideas to increase the scalability, such as using building blocks to automatically generate portions of the course of action.

5.2.2 Hidden Markov Models

A hidden Markov model (HMM) is a Bayesian network where the current state is dependent upon only the previous state. Each edge in an HMM is modeled by a transition probability. They are assigned in the same way discussed in the previous section. Using the ideas from [24], we apply them to cyber attacks. Figure 133 shows a potential course of action. This course of action is converted to an HMM in 14. Note that there are 9 states in the HMM which was generated from a model with only 5. As the course of action complexity increases, the HMM will exponentially increase in size, thus creating a significantly larger number of transition probabilities.

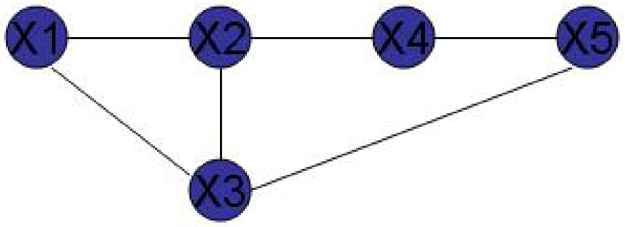


Figure 13: Example Guidance Template

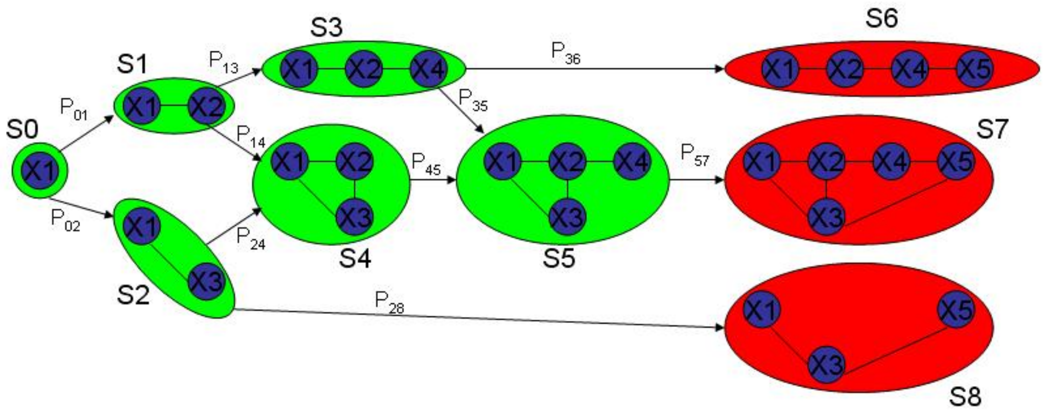


Figure 14: Example HMM derived from Figure 1

5.2.3 Potential Drawbacks

The correctness of stochastic models heavily depends on the accuracy of the probabilities. The following is a list of reasons why it may not be possible to model the probabilities accurately enough to perform a worthwhile impact assessment.

- *Complexity*
  As evidenced by the HMM example, the number of states grows at a very fast rate as new states in the course of action are added. This creates a large number of edges in which to assign probabilities. An increased complexity could lead to a high variance among probabilities from different subject matter experts.

- *High Variance*
  Subject matter experts are usually asked to assign the probabilities. However, it could be likely that if a number of different experts are asked to assign these probabilities, there could be a large variance of probabilities among the experts, thus making the average probabilities inaccurate. The wide variance could be due to the aforementioned complexity of the model. It is also possible that the probabilities will widely vary among different networks.

- *Incomplete Training Data*
  While training data could be used to train the transition probabilities, the training data must be representative of the set of all possible data. Due to the minimal data available for cyber attacks, the training data will be incomplete, thus making the model inaccurate.

- *Dynamically Changing Probabilities*
  New vulnerabilities and exploits will potentially alter the probabilities. This implies that the model needs to dynamically change. However, if there is a dynamic model, it is possible that hackers could artificially inflate some probabilities by performing a large number of decoy attacks. The inflation of these probabilities could bias the model in such a way that a likely attack will not be interpreted as a likely attack.

5.3 Threat Assessment of Network Data and Information

Because of the drawbacks to stochastic techniques, we propose a deterministic approach to assess the impact and threat of a cyber attack. Any probabilities that may be associated with any transitions are ignored, so it is assumed that all transitions are equally likely. The proposed model assumes the worst case capability and intent of the hacker – an advanced malicious hacker. Modeling and estimating the capability of the hacker is outside of the scope of this project, as it would require extensive studies involving the behavior of a hacker. The intent of the hacker was also determined to be too unpredictable to accurately model. The primary focus of this model is the opportunity of the hacker. This model is a good building block that could eventually be extended to incorporate probabilities, multiple hacker capabilities and intent.

5.4 Inputs

The following inputs are necessary for this framework (the details of the inputs will be discussed as the algorithm is presented):

- Information hierarchy and associated criticalities
- Guidance template of possible attacks independent of targets
- Logical network topology
- Mapping of attacks/privileges to information nodes
- Mapping of targets to information nodes
- Set weights (λ_1, λ_2, λ_3, λ_4)

5.5 Information Hierarchy

The information hierarchy is a directed graph where each node represents a piece of information that a hacker needs to obtain even more information. The edges represent the order in which information needs to be compromised. For example, if an edge is defined from node A to node B, it implies that the information node A represents must be compromised before the information represented by node B can be compromised.

Each piece of information is mapped to all of the targets that can be used to access the information as well as all of the attacks that could be used to obtain the information. Each node is also assigned a criticality representative of the impact of that information being compromised. The targets are related by the logical topology and the attacks are related by a guidance template. The logical topology is defined by the order in which machines must be compromised (this will be a very dense graph for the internal network). The guidance template is simply a relation of attack types to define a potential course of action for the hacker.

Figure 15 shows the information hierarchy and how it relates to the logical topology and guidance template. To reduce the number of arrows shown, each box in the information hierarchy indicates that each node is mapped to the machine (the nodes defined in the logical topology) pointing to the box. The color of the information node illustrates what attack it maps to.

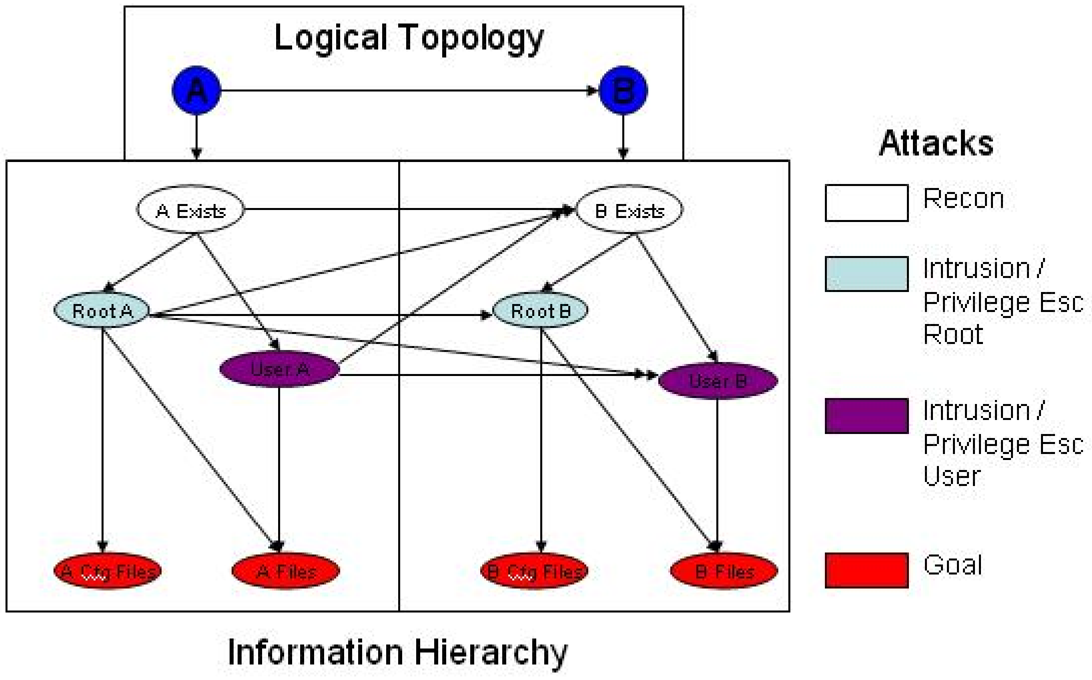


Figure 15: Simple example of our framework

Each machine has a minimum of three levels in the information hierarchy. The first level indicates that the hacker must know that the machine exists and potentially other machine attributes. The second level represents the privilege levels. Any user account could potentially be escalated to root privileges by the hacker. The third level is the actual data stored on the machine. Note that user or root access to a machine allows access an adjacent machine based on the logical topology. This may not be true in all cases, but edges could always be added or removed as necessary. This is just a simple example to illustrate the relationship between the guidance template, information hierarchy and logical topology.

5.6 Impact Assessment

Before describing the mathematical formulation of the impact assessment, we first define notation. Let *G* define the total number of attacks that have occurred. Let A_i^* denote the type of attack of the i^{th} attack. Let M_i^* denote the target of the i^{th} attack. From this $\mathbf{A}^*$ (the set of attacks that have occurred) is defined as:

$$\mathbf{A}^* = \bigcup_{i=1}^{G} A_i^* \qquad (12)$$

Similarly, $\mathbf{M}^*$ is the set of machines that have been attacked:

$$\mathbf{M}^* = \bigcup_{i=1}^{G} M_i^* \tag{13}$$

Let **A**(**X**), **I**(**X**), and **M**(**X**) denote the attacks, information, and machines, respectively, adjacent to the set **X** based on the mappings previously described.

Let the estimate of the set of currently compromised information be denoted by $\mathbf{I}^*$. It is defined as:

$$\mathbf{I}^* = \bigcup_{i=1}^{G} \left(\mathbf{I}(A_i^*) \cap \mathbf{I}(M_i^*) \right) \tag{14}$$

Information is considered compromised if an attack type is mapped to a piece of information and the target contains that information.

5.7 Threat Assessment

The threat of a node *i* is denoted as *t(i).* The impact assessment indicated an estimate of the pieces of information that the hacker currently has. Therefore, the threat of each piece of information in $\mathbf{I}^*$ can be considered to be one, i.e:

$$t(I_i^*) = 1, 1 \le i \le |\mathbf{I}^*| \tag{15}$$

It is now of interest to calculate the threat of the next possible pieces of information to be compromised. Recall that the information hierarchy is defined such that the parent nodes must be compromised before the children nodes. Therefore, the threat only needs to be assessed for the set $\mathbf{I}(\mathbf{I}^*)$. There are four different sets used to assess the threat using this algorithm:

$$\mathbf{S}_1 = \mathbf{I}(\mathbf{M}^*) \tag{16}$$

$$\mathbf{S}_2 = \mathbf{I}(\mathbf{M}(\mathbf{M}^*)) \tag{17}$$

$$\mathbf{S}_3 = \mathbf{I}(\mathbf{A}^*) \tag{18}$$

$$\mathbf{S}_4 = \mathbf{I}(\mathbf{A}(\mathbf{A}^*)) \tag{19}$$

$\mathbf{S}_1$ is the set of information associated with all machines that have been attacked. Since those machines have been attacked, they are at risk of being attacked again. $\mathbf{S}_2$ is the set of information associated with machines adjacent to the ones that have been attacked. These machines are at a lower risk since they have not been attacked, but they can be attacked next because of the logical topology. $\mathbf{S}_3$ is the information that could be compromised by the set of attacks that already have occurred. Since the attack has already occurred, this type of attack can possibly occur again. $\mathbf{S}_4$ is the information that could be compromised by the set of next attacks that the hacker could perform.

An attack is more likely to occur on a machine in $\mathbf{M}^*$ than a machine in $\mathbf{M}(\mathbf{M}^*)$ so a higher weight is assigned to $\mathbf{S}_1$ than to $\mathbf{S}_2$. The similar case is true for $\mathbf{S}_3$ and $\mathbf{S}_4$. The weight of $\mathbf{S}_i$ will be λ where the λ's are defined such that:

$$\sum_{k=1}^{4} \lambda_k = \beta, 0 \leq \beta \leq 1 \tag{20}$$

where β is defined as the maximum threat of a non-compromised node.

To assess the threat of a piece of information, the following indicator function is defined to denote if node i is contained within set $\mathbf{S}$:

$$m(i, \mathbf{S}) = \begin{cases} 0, i \notin \mathbf{S} \\ 1, i \in \mathbf{S} \end{cases} \tag{21}$$

The threat for a node, i, contained within $\mathbf{I}(\mathbf{I}^*)$ is defined as:

$$t(i) = \sum_{k=1}^{N} \lambda_k m\left(i, \mathbf{S_k}\right) \tag{22}$$

In other words, the threat of the node is the sum of the weights corresponding to the sets that the node is contained in. The maximum value of the function is incurred when the node is contained within all four sets, so the threat level will be β. For completeness:

$$t(i) = 0, i \notin \left(\mathbf{I}^* \cup \mathbf{I}(\mathbf{I}^*)\right) \tag{23}$$

Equations 15, 22, and 23 can be combined to create an equation for the threat of any node in the information hierarchy:

$$t(i) = \max\left(m(i, \mathbf{I}^*), m(i, \mathbf{I}(\mathbf{I}^*)) \sum_{k=1}^{N} \lambda_k m\left(i, \mathbf{S_k}\right) \right) \tag{24}$$

5.8 Generalization of Framework

The framework just presented could be generalized to other domains. The type of attack and machine are two attributes of an attack. Other domains may have more than two attributes. Our approach will now be generalized to other domains by expanding the number of possible attributes. Let K be the number of categories of attack attributes. Let each category be denoted by C_k where $1 \leq k \leq K$. $\mathbf{C}^*_{(\mathbf{k,i})}$ denotes the attack attribute(s) in the kth category of the ith attack. Based on the $\mathbf{C}^*$'s, we now define J sets, each identified by $\mathbf{S_j}$ where $1 \leq j \leq J$. Every $\mathbf{S_j}$ has a corresponding λ_j which is the weight of the set. The λ_j 's are defined similar to Equation 20:

$$\sum_{j=1}^{J} \lambda_j = \beta, 0 \le \beta \le 1 \tag{25}$$

5.9 Impact Assessment

We define the impact assessment similar to Equation 14:

$$\mathbf{I}^* = \bigcup_{i=1}^{G} \bigcap_{k=1}^{K} \mathbf{I}\left(\mathbf{C}^*_{(\mathbf{k},\mathbf{i})}\right) \tag{26}$$

5.10 Threat Assessment

We define the threat assessment similar to Equation 24:

$$t(i) = \max\left(m(i, \mathbf{I^*}), m(i, \mathbf{I}(\mathbf{I^*})) \sum_{j=1}^{N} \lambda_j m\left(i, \mathbf{S_j}\right) \right) \tag{27}$$

5.11 Future Extensions

This approach can lead to a very large information hierarchy, thus making it unwieldy for large networks. However, this information hierarchy could be created defining generic templates for workstations and servers and generating only a portion of the information hierarchy dynamically. This will reduce the spatial complexity of our approach and will filter out the portions of the information hierarchy that are not being threatened. We plan to implement this change soon.

We also plan to modify the approach slightly to use INFERD to not only generate the information hierarchy, but to also assess the threat. The impact and threat equations we used could be converted to INFERD's architecture. Another extension includes factoring in multiple attack tracks to analyze the threat. If more than one attack occurs on the same node, the neighboring nodes should be at a higher threat since there are more avenues in which to compromise the data. Finally, we plan to simulate this algorithm with realistic data to analyze the effectiveness of the proposed algorithm.

6 Simulation Environment

Information fusion is the process of associating, correlating, and combining data and information from single or multiple sources to estimate parameters, characteristics, and behaviors of a system for the purposes of analysis or decision support [4]. Figure 16 illustrates the application of information fusion to a system. The ground truth is the actual status of the system. From the ground truth, a set of data or information can be sensed and passed to an information fusion process. The fused information is passed to a decision maker that may take some action on the system in attempt to change the system status.

Some of the most difficult aspects of developing information fusion methods are validation and evaluation. The validation and evaluation processes both require data for testing and experimentation. In some cases, the systems to which the information fusion process is to be used are readily available so direct experimentation can take place. However, in many applications the systems for which the information fusion processes are being designed do not exist, may be destructive, or may be cost prohibitive to set up. In these cases, simulation provides a good alternative.

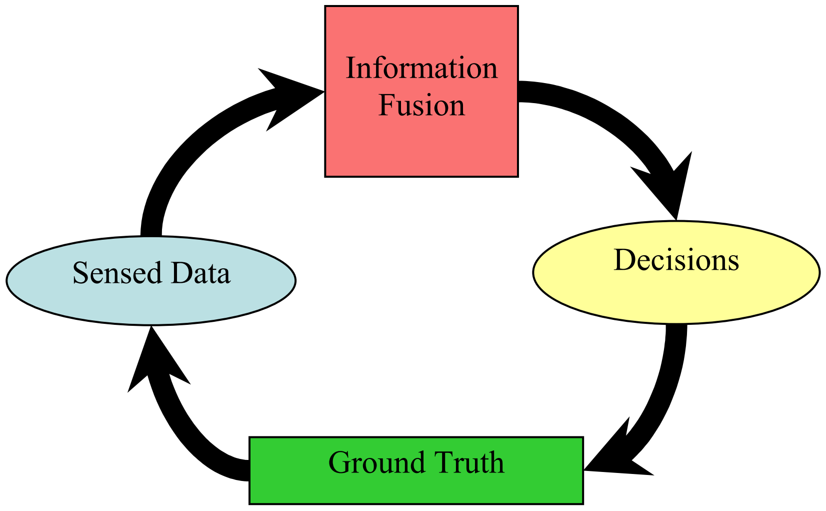


Figure 16: Information Fusion Applied to a System

For example, in the context of cyber security, situational awareness and threat assessment tools including information fusion techniques are being developed to aid systems administrators in identifying and analyzing cyber attacks on computer networks [14]. These tools work by primarily processing alerts produced by intrusion detection systems (sensors) on the computer network. To test and evaluate these tools, physical computer networks have been set up to perform experiments from which data is collected. As an alternative, a simulation modeling method and software is developed to generate synthetic data.

Simulation is often used to model systems with complex behavior and subsequently used as a basis for testing algorithms and methods including information fusion methods. For example, Lee et al. [9] and Nicol et al. [11] present simulation modeling methods for simulating computer network traffic at the packet level. Although simulating the flow and processing of packets in the computer network is possible (potentially billions of packets per day), only a small fraction of the packets cause alerts to be produced by the intrusion detection system which in turn would be used by the information fusion tools. Furthermore, modeling a system at this level of detail requires great amounts of time and effort for modeling as well as requiring large amounts of computer processing time for simulating "good" packets. As an alternative to modeling the details of packet flow in a network, this work presents a simulation model for simulating the behavior of the intrusion detection system by producing simulated alerts representative of malicious cyber attacks and non-malicious network activity based on the user's specification. Consequently, the user can efficiently construct scenarios of various computer networks and cyber attacks and generate the corresponding alerts.

6.1 Problem Statement

Data analysis and decision support systems, such as information fusion tools, require experimental situations and data to be tested during development and for qualification. In many domains, obtaining experimental data from physical systems could be dangerous, destructive, or expensive. The goal of this work is to create a simulation framework for generating synthetic data for the purposes of experimentation and testing. In particular, the objective is to create a Cyber Attack Simulator that will take as input the network configuration and information about cyber attacks on a network and create data sets representative of alerts produced by intrusion detection systems as well as a description of the ground truth.

6.2 Background and Related Work

This work is based in the need for testing situational awareness tools that are being developed to detect and analyze attacks on computer networks. Since conducting cyber attack experiments on computer systems that contain critical data is very undesirable, several alternatives have been used. One alternative consists of setting up a physical computer network absent of any critical data, performing cyber attacks on the network, and collecting data from intrusion detection systems. A second alternative consists of generating synthetic data through the use of simulation.

These two approaches have varying degrees of requirements, capabilities, and limitations. The physical computer network requires the physical machines, networking, and IDS components. Consequently, conducting experiments on various network configurations involving different machines, servers, routing systems, IDS sensors, etc. requires reconfiguration of the network and setting up the network to produce the desired network activity and cyber attacks. The advantage of using the physical network is that the data produced is from a real network as opposed to an abstract representation. This also has some disadvantages in that it is impossible to replicate the experiment exactly (if so desired) and the data produced is difficult to validate to ensure all desired information is accounted for in the ground truth. Since physical networks are not perfectly reliable, data can be missed, processed incorrectly, etc.

The simulation approach requires knowledge of the operation of the desired network and its operation. This information must be captured by the simulation model to represent the behavior of the network. However, as discussed briefly in the introduction, the level of detail included in the model will depend on the goal of the simulation. In this case, the packet level information and computer network traffic details are not needed, so the simulation can be constructed at a higher level to produce alerts caused by cyber attacks and harmless network traffic. Once the framework of the model has been established, various network configurations can be efficiently created and experiments can be conducted with various attack scenarios. Since the simulation experiments are controlled, they can be repeated exactly and all ground truth information is known.

6.3 General Overview of the Simulation Model for Cyber Attack Data Generation

A simulation model for generating cyber attack data is developed which is to be used for testing cyber situational awareness and analysis tools. To keep the focus on accurately modeling cyber attacks and the resulting alert messages, a commercial simulation package, ARENA, was used.

The capabilities of the simulation model allow a user to construct a computer network consisting of machines that can be grouped in subnets though switches that are modeled as connectors. The subnets can be connected to form a complete network. The machines can also be categorized in terms of their accessibility from outside the network. In addition, IDS sensors can be placed on the connectors to model network IDS sensors and on the machines to model host based IDS sensors. Each IDS sensor produces an output file containing the alerts produced during the simulation run.

The user can create attack scenarios that include the ability to specify a sequence of attack actions over a period of time. A scenario can currently consist of up to 10 attacks with up to 30 steps (actions) for each attack. The user can specify a constant time between steps in the attack, the mean time between steps in the attack, or the mean time for the entire attack. If the mean times are specified the resulting time is generated randomly form the exponential distribution during the simulation. For each attack step, the user can specify the attacker IP address to be one of the computers in the network or the simulation can generate an IP for a computer outside the network. Once the attacker IP is specified, the user can select a target IP from a subset of computers within the network with which the attacking IP can communicate. The user can then specify the action for each step of the attack. The 2,237 current attack actions are categorized into 5 major groups and 23 subgroups. The user can specify a specific action; or the simulation will randomly sample from the actions within a group or subgroup depending on the level specified by the user. Also, the user can specify the probability of success of each attack step. The simulation will then randomly sample to determine whether the step is successful during the simulation run. If the attack step fails, the step will be repeated until success is achieved.

In addition to attacks, the user can specify, the rate at which non-malicious alerts (noise) is generated, as well as the probability of noise alerts corresponding to each of the action categories.

Once the scenario has been created, the information is saved in a file for future use. The simulation is then run, and the attack scenario is executed. The output of the simulation includes a file representative of the ground truth that contains the actions generated for each attack and the time the action occurred. In addition, an output file containing IDS alerts is produced for each IDS specified in the model. These files containing IDS alerts are intended to be used to test the situational awareness and analysis tools.

An interface has been created to enable the user to easily interact with the simulation models. For additional information, please see the software user's guide in Appendix A.

6.4 Simulation Methodology

Although the commercial simulation software, ARENA, is used to develop the synthetic data set generator for cyber attacks, a significant amount of customization was required to obtain the completed simulation tool. In this section, the general modeling approach is discussed with the use of standard ARENA constructs followed by a discussion of the customization.

6.4.1 General Modeling Approach

The general modeling approach for representing the computer network and cyber attacks is to model the individual cyber attacks as entities, and the locations of the machines within the network are modeled as stations. Figure 17 shows the model logic for entity creation and initialization. One entity is created at the beginning of the simulation to represent each attack. Each entity is assigned a unique attack identification number. Then the entity executes the VBA block that samples the necessary attack information which depends on the user specifications. The assign block following the VBA block is used to assign the information for the first attack step to entity attributes. The FindJ block is used to determine the station that corresponds to the IP address of the target machine in the first attack step, and the station number is assigned to an attribute. Then the entity is delayed until the first step in the attack is specified to start. This delay can be constant or can be sampled randomly from an exponential distribution depending on the user's input. Finally, the entity is routed to the station corresponding to the target IP address for the first attack step.

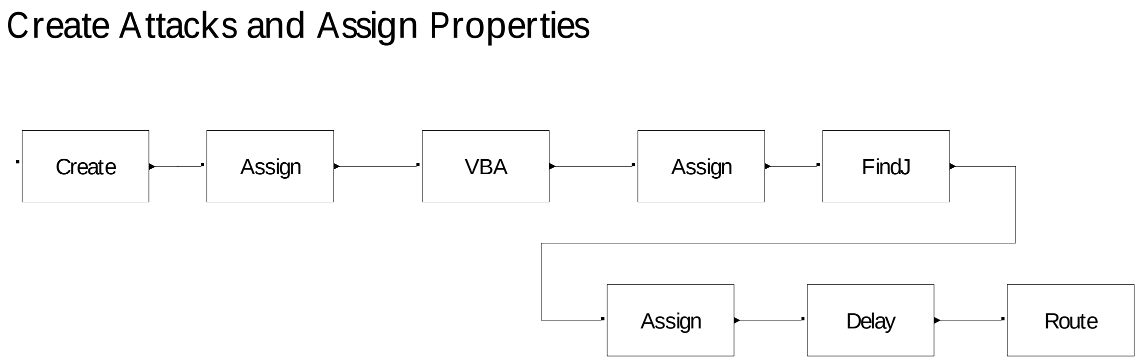


Figure 17: Simulation Model Logic for Creating Entities Representative of Cyber Attacks

Figure 18 shows the generic station submodel that represents each of the machine locations in the computer network. When entities (attacks) are routed to the station module, they enter the branch block to send the entity to the next appropriate blocks for processing. The first two branches route the entity to the module logic shown in Figure 18 to print the corresponding IDS alerts. The third branch sends the entity to another branch block where the success of the attack step is evaluated by sampling from a uniform distribution on (0,1) and comparing this number with the probability of success specified by the user. If the step fails, the necessary attack step information is sampled via the VBA block, the attack information is assigned to attributes, the target IP station is determined, the entity is delayed, and the entity is then routed to the appropriate station similar to the sequence of action executed in Figure 17. If the attack step succeeds, the attack step number is incremented by one, and the next step in the attack is executed. This process is repeated until the last step in the attack is executed successfully. At this point, the number of completed attacks is tallied, and the entity is disposed.

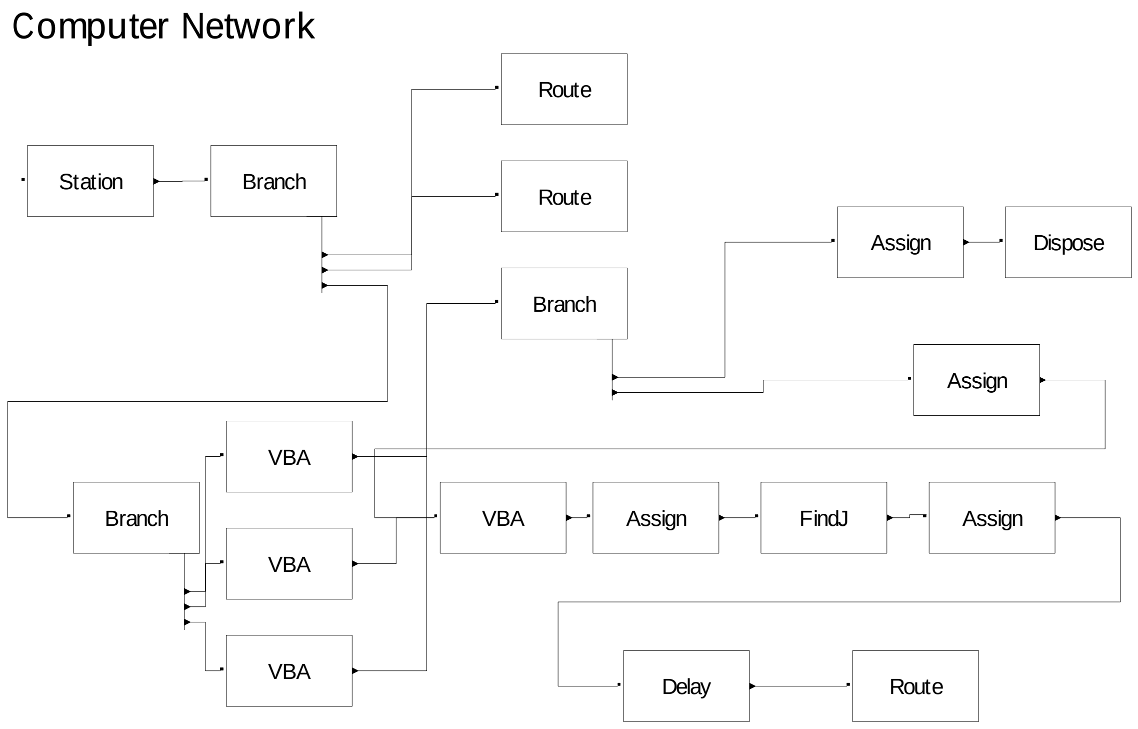


Figure 18: Generic Station Submodel Representing Network Computers

Figure 19 shows the simulation model logic used to write the ground truth and IDS alerts to output files. The Ground Truth logic produces records in files that contain all of the IDS alerts that would be produced if the IDS sensors covered the entire network. The IDS Alerts logic produces records in files that contain only the IDS alerts that are produced by the IDSs which are dependent on the location of the IDS in the network.

Figure 20 shows the simulation model logic for modeling the creation, execution, and recording of noise alerts. The occurrence rate of noise alerts are specified by the user, and generated via a Poisson arrival process. The VBA blocks are used to record the ground truth and IDS alert files.

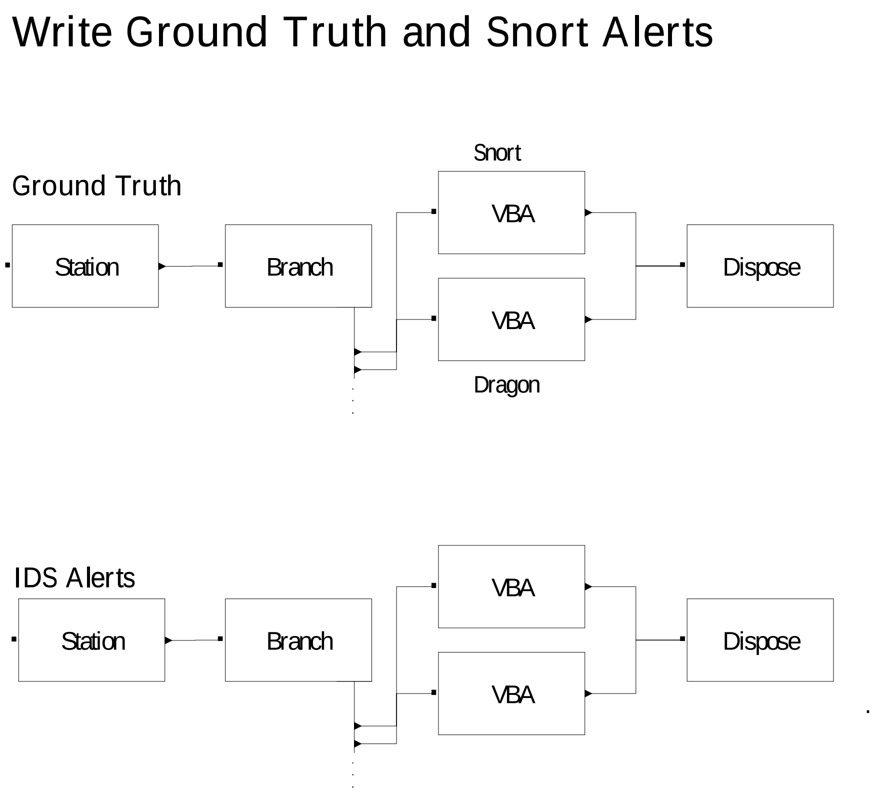


Figure 19: Simulation Model Logic for Writing Ground Truth and IDS Alert Files

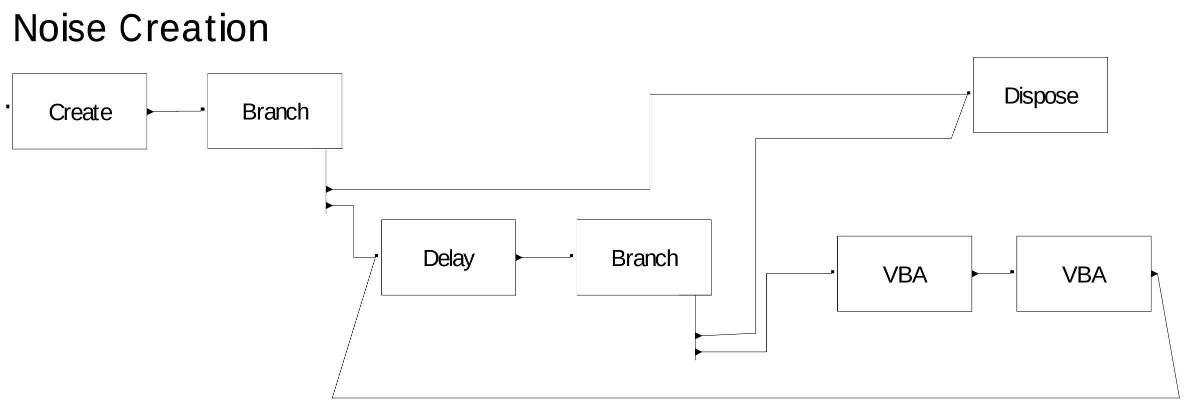


Figure 20: Simulation Model Logic for Producing Noise Alerts

6.4.2 Customization of the Simulation Software

Significant customization of the ARENA simulation software was required to model computer networks and cyber attacks. The reasons for the need to customize the software include the fact that a simulation framework was being designed where the user could efficiently set up a network and specify the desired attacks. One of the major aspects of the customization was to create a template containing icons that represented machines and connectors to represent the machines and switches in computer network, respectively. Figure 21 shows the template that was created (see Kelton et al. 2004 for information on customization of ARENA). From this template, the user can drag and drop icons into the ARENA simulation environment to create a representation of the computer network. Using the connection tool in ARENA, the user can specify the connections of machines to connectors to model subnets and network connectivity. Figure 22 shows a sample computer network created using the Network template.

Within the icon modules, the user can specify network information including the IP address of each machine, whether the machine can be accessed from an external source, and the user can specify whether the machine is to be monitored by a host based IDS. The user can also specify their own descriptive name to distinguish between machines in the network. Within the connector icon module, the user can specify the connectivity to among subnets as well as specify whether a network IDS should be turned on.

In addition to the template, custom forms were created in visual basic to serve as input screens for the user to specify details about the attacks and noise that are to be simulated. Furthermore, many custom features were placed in the input menus to allow for ease of modeling and reuse of input parameters. Finally, a user interface was created to view the output files created for the attack alert and ground truth during the simulation. For additional details about the user interfaces, please see the user manual in Appendix A.

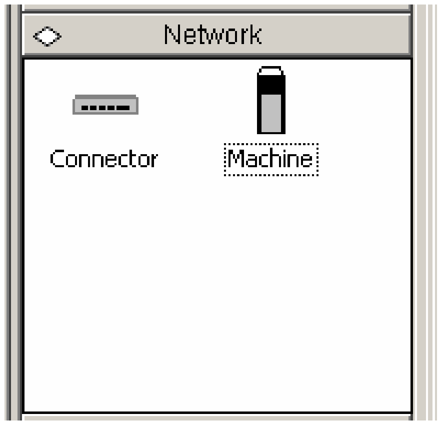


Figure 21: Network Template

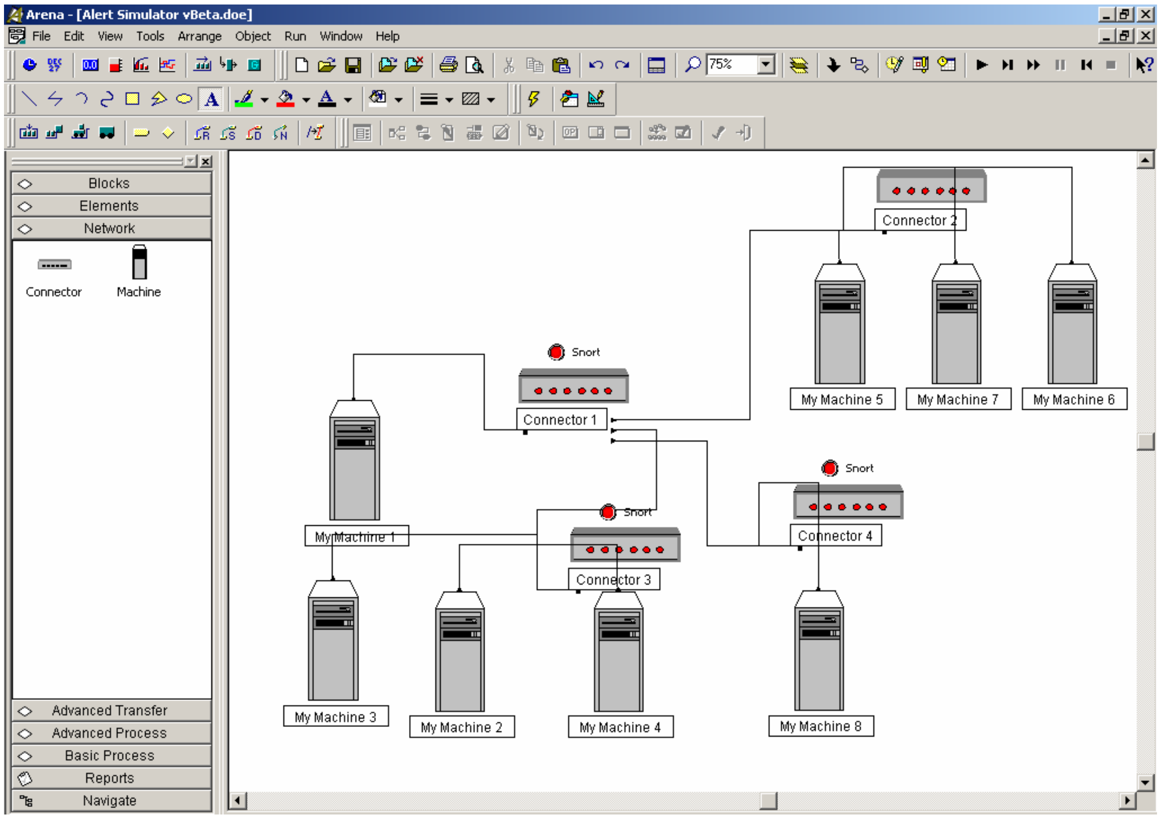


Figure 22: Sample Computer Network Consisting of Four Subnets

6.5 Example

In this section, an example attack scenario on a computer network is presented.

Attack Description

The goal of this attack is to create a denial of service on a machine on the internal network. The network diagram is shown in Figure 23. Information is gathered about the external network, and then the VPN server is penetrated. The server is then used as a stepping stone to reach the target machine. The following are the steps of the attack:

1. Enumeration on VPN server from outside of the network attempting to get user passwords. Succeeded on first attempt. Step was encoded.
2. Intrusion at the user level on the VPN server from outside of network. Succeeded on first attempt and attacker gained access to server. Step was encoded.
3. Backdoor left on the VPN server for future access if necessary. Succeeded on first attempt. Step was encoded.
4. Reconnaissance on machine in subnet with snort sensor, specifically ICMP Ping from VPN server. Succeeded on first attempt.
5. Intrusion at the user level on machine 100.10.20.1 from VPN server. Succeeded on first attempt.

6. Denial of service enacted on 100.10.20.1 from VPN server. Succeeded on first attempt. Step was encoded.

The computer network created using the Cyber Attack Simulator is shown in Figure 23. A summary of the network is as follows:

- 1 main web-server;
- 4 main subnet domains;
- 3 subnet domains have further subnets attached;
- Only one external machine; and
- Red dots indicate snort sensor presence

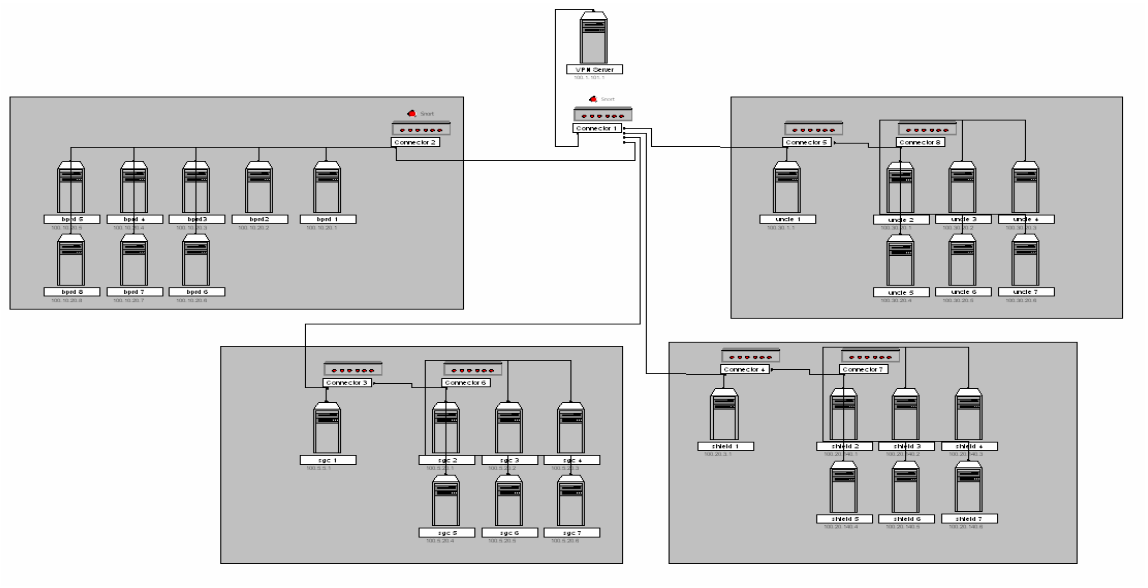
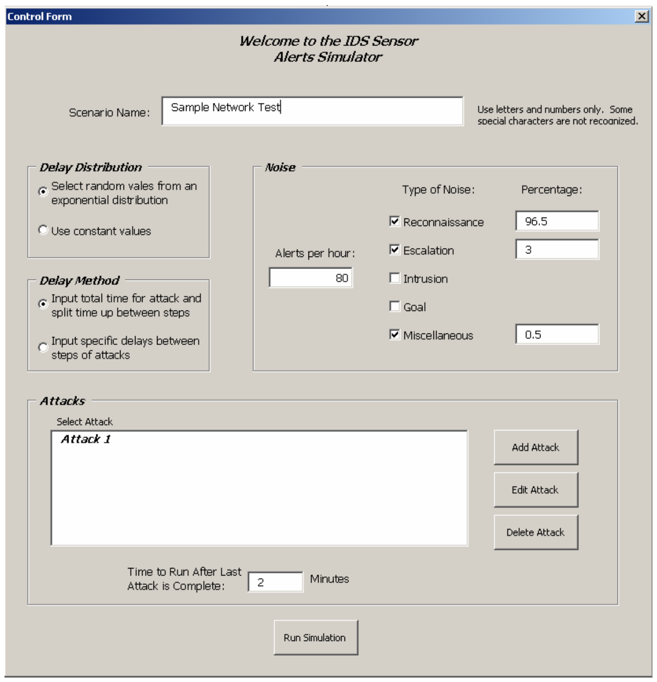


Figure 23: Sample Network

Figures 24 and 25 illustrate the user interface and specifications entered for the attack. These specifications include the following details:

- The name of the scenario is Sample Network Test;
- Delay distribution is exponential, total time for attack is broken up between steps;
- There are 80 noise alerts per hour on average. 96.5% of the noise is reconnaissance, 3% is escalation and 0.5% is miscellaneous;
- There is one attack called Attack 1; and
- The simulation will run for two minutes after the last attack is complete.


Control Form

Welcome to the IDS Sensor Alerts Simulator

Scenario Name: Sample Network Test

Use letters and numbers only. Some special characters are not recognized.

Delay Distribution
- [✓ Select random vales from an exponential distribution]
- Use constant values

Delay Method
- [✓ Input total time for attack and split time up between steps]
- Input specific delays between steps of attacks

Noise

Alerts per hour: 80

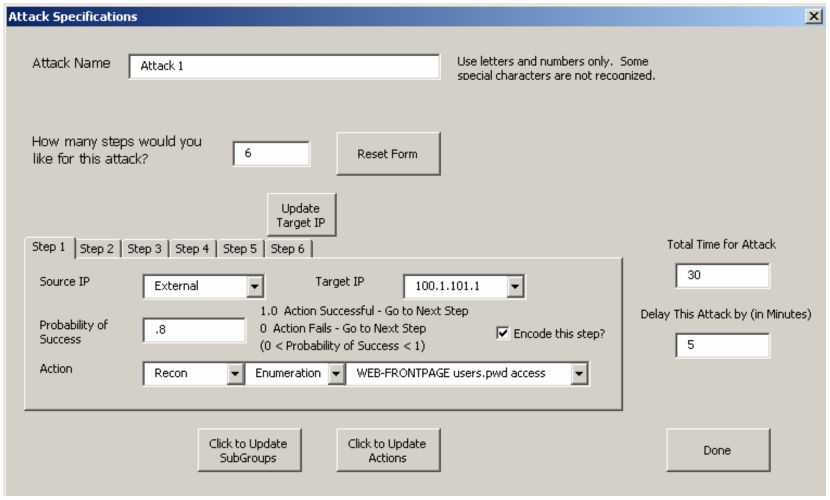
Type of Noise:	Percentage:
☑ Reconnaissance	96.5
☑ Escalation	3
☐ Intrusion	
☐ Goal	
☑ Miscellaneous	0.5

Attacks

Select Attack: *Attack 1*

Time to Run After Last Attack is Complete: 2 Minutes

Figure 24: Scenario Specification Dialog Box for Example


Attack Specifications

Attack Name: Attack 1

Use letters and numbers only. Some special characters are not recognized.

How many steps would you like for this attack? 6

Step 1 | Step 2 | Step 3 | Step 4 | Step 5 | Step 6

Source IP: External

Target IP: 100.1.101.1

Probability of Success: .8

1.0 Action Successful - Go to Next Step
0 Action Fails - Go to Next Step
(0 < Probability of Success < 1)

☑ Encode this step?

Action: Recon | Enumeration | WEB-FRONTPAGE users.pwd access

Total Time for Attack: 30

Delay This Attack by (in Minutes): 5

Figure 25: Attack Dialog Box for Example

The simulation results in the output files shown in Figures 26, 27, and 28 being produced. The file in Figure 26 describes the attacks in a format similar to the way it was created. The first three items displayed are the action group, subgroup and the specific action performed. If the action was encoded, it is indicated after the action. Next are the source and target IP address, followed by the probability of success. Finally, the file indicates whether the action succeeded or failed.

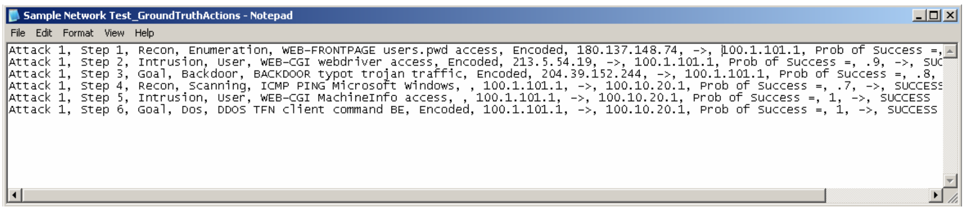

```
Attack 1, Step 1, Recon, Enumeration, WEB-FRONTPAGE users.pwd access, Encoded, 180.137.148.74, ->, 100.1.101.1, Prob of Success =,
Attack 1, Step 2, Intrusion, User, WEB-CGI webdriver access, Encoded, 213.5.54.19, ->, 100.1.101.1, Prob of Success =, .9, ->, SUC
Attack 1, Step 3, Goal, Backdoor, BACKDOOR typot trojan traffic, Encoded, 204.39.152.244, ->, 100.1.101.1, Prob of Success =, .8,
Attack 1, Step 4, Recon, Scanning, ICMP PING Microsoft Windows, , 100.1.101.1, ->, 100.10.20.1, Prob of Success =, .7, ->, SUCCESS
Attack 1, Step 5, Intrusion, User, WEB-CGI MachineInfo access, , 100.1.101.1, ->, 100.10.20.1, Prob of Success =, 1, ->, SUCCESS
Attack 1, Step 6, Goal, Dos, DDOS TFN client command BE, Encoded, 100.1.101.1, ->, 100.10.20.1, Prob of Success =, 1, ->, SUCCESS
```

Figure 26: Ground Truth Actions

The file in Figure 27 displays the alert messages for each action in the ground truth actions file including all successes and failures.

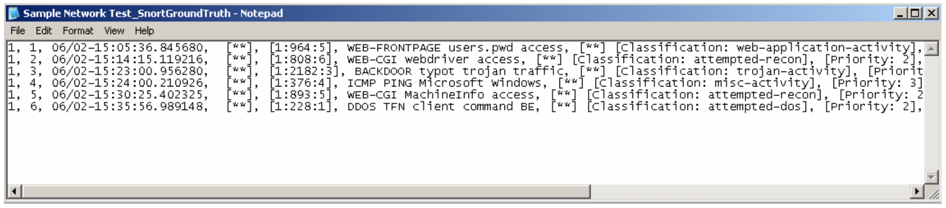

```
1, 1, 06/02-15:05:36.845680,   [**], [1:964:5], WEB-FRONTPAGE users.pwd access, [**] [Classification: web-application-activity],
1, 2, 06/02-15:14:15.119216,   [**], [1:808:6], WEB-CGI webdriver access, [**] [Classification: attempted-recon], [Priority: 2],
1, 3, 06/02-15:23:00.956280,   [**], [1:2182:3], BACKDOOR typot trojan traffic, [**] [Classification: trojan-activity], [Priorit
1, 4, 06/02-15:24:00.210926,   [**], [1:376:4], ICMP PING Microsoft Windows, [**] [Classification: misc-activity], [Priority: 3]
1, 5, 06/02-15:30:25.402325,   [**], [1:893:5], WEB-CGI MachineInfo access, [**] [Classification: attempted-recon], [Priority: 2
1, 6, 06/02-15:35:56.989148,   [**], [1:228:1], DDOS TFN client command BE, [**] [Classification: attempted-dos], [Priority: 2],
```

Figure 27: Ground Truth Alerts

The file in Figure 28 shows all alerts created by noise and attack actions which the SnortNetwork_1 IDS sensor would be able to detect given its location in the network.

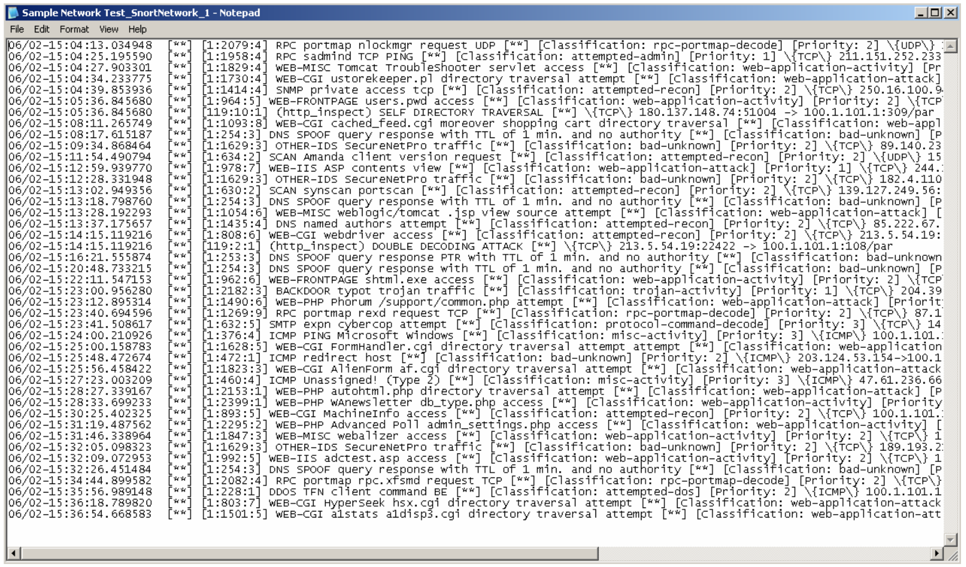

```
06/02-15:04:13.034948   [**]  [1:2079:4] RPC portmap nlockmgr request UDP [**] [Classification: rpc-portmap-decode] [Priority: 2] \{UDP\}
06/02-15:04:25.195590   [**]  [1:1958:4] RPC sadmind TCP PING [**] [Classification: attempted-admin] [Priority: 1] \{TCP\} 211.151.252.233
06/02-15:04:27.903301   [**]  [1:1829:4] WEB-MISC Tomcat TroubleShooter servlet access [**] [Classification: web-application-activity] [Pr
06/02-15:04:34.233775   [**]  [1:1730:4] WEB-CGI ustorekeeper.pl directory traversal attempt [**] [Classification: web-application-attack]
06/02-15:04:39.853936   [**]  [1:1414:4] SNMP private access tcp [**] [Classification: attempted-recon] [Priority: 2] \{TCP\} 250.16.100.9
06/02-15:05:36.845680   [**]  [1:964:5] WEB-FRONTPAGE users.pwd access [**] [Classification: web-application-activity] [Priority: 2] \{TCP
06/02-15:05:36.845680   [**]  [119:10:1] (http_inspect) SELF DIRECTORY TRAVERSAL [**] \{TCP\} 180.137.148.74:51004 -> 100.1.101.1:309/par
06/02-15:08:11.265749   [**]  [1:1093:8] WEB-CGI cached_feed.cgi moreover shopping cart directory traversal [**] [Classification: web-appl
06/02-15:08:17.615187   [**]  [1:254:3] DNS SPOOF query response with TTL of 1 min. and no authority [**] [Classification: bad-unknown] [P
06/02-15:09:34.868464   [**]  [1:1629:3] OTHER-IDS SecureNetPro traffic [**] [Classification: bad-unknown] [Priority: 2] \{TCP\} 89.140.23
06/02-15:11:54.490794   [**]  [1:634:2] SCAN Amanda client version request [**] [Classification: attempted-recon] [Priority: 2] \{UDP\} 15
06/02-15:12:59.939770   [**]  [1:978:7] WEB-IIS ASP contents view [**] [Classification: web-application-attack] [Priority: 1] \{TCP\} 244.
06/02-15:12:28.331948   [**]  [1:1629:3] OTHER-IDS SecureNetPro traffic [**] [Classification: bad-unknown] [Priority: 2] \{TCP\} 182.4.110
06/02-15:13:02.949356   [**]  [1:630:2] SCAN synscan portscan [**] [Classification: attempted-recon] [Priority: 2] \{TCP\} 139.127.249.56:
06/02-15:13:18.798760   [**]  [1:254:3] DNS SPOOF query response with TTL of 1 min. and no authority [**] [Classification: bad-unknown] [P
06/02-15:13:28.192293   [**]  [1:1054:6] WEB-MISC weblogic/tomcat .jsp view source attempt [**] [Classification: web-application-attack] [
06/02-15:13:37.175657   [**]  [1:1435:4] DNS named authors attempt [**] [Classification: attempted-recon] [Priority: 2] \{TCP\} 85.222.67.
06/02-15:14:15.119216   [**]  [1:808:6] WEB-CGI webdriver access [**] [Classification: attempted-recon] [Priority: 2] \{TCP\} 213.5.54.19:
06/02-15:14:15.119216   [**]  [119:2:1] (http_inspect) DOUBLE DECODING ATTACK [**] \{TCP\} 213.5.54.19:22422 -> 100.1.101.1:108/par
06/02-15:16:21.555874   [**]  [1:253:3] DNS SPOOF query response PTR with TTL of 1 min. and no authority [**] [Classification: bad-unknown
06/02-15:20:48.733215   [**]  [1:254:3] DNS SPOOF query response with TTL of 1 min. and no authority [**] [Classification: bad-unknown] [P
06/02-15:22:11.547153   [**]  [1:962:6] WEB-FRONTPAGE shtml.exe access [**] [Classification: web-application-activity] [Priority: 2] \{TCP
06/02-15:23:00.956280   [**]  [1:2182:3] BACKDOOR typot trojan traffic [**] [Classification: trojan-activity] [Priority: 1] \{TCP\} 204.39
06/02-15:23:12.895314   [**]  [1:1490:6] WEB-PHP Phorum /support/common.php attempt [**] [Classification: web-application-attack] [Priorit
06/02-15:23:40.694596   [**]  [1:1269:9] RPC portmap rexd request TCP [**] [Classification: rpc-portmap-decode] [Priority: 2] \{TCP\} 87.1
06/02-15:23:41.508617   [**]  [1:632:5] SMTP expn cybercop attempt [**] [Classification: protocol-command-decode] [Priority: 3] \{TCP\} 14
06/02-15:24:00.210926   [**]  [1:376:4] ICMP PING Microsoft Windows [**] [Classification: misc-activity] [Priority: 3] \{ICMP\} 100.1.101.
06/02-15:25:00.158783   [**]  [1:1628:5] WEB-CGI FormHandler.cgi directory traversal attempt attempt [**] [Classification: web-application
06/02-15:25:48.472674   [**]  [1:472:1] ICMP redirect host [**] [Classification: bad-unknown] [Priority: 2] \{ICMP\} 203.124.53.154->100.1
06/02-15:25:56.458422   [**]  [1:1823:3] WEB-CGI AlienForm af.cgi directory traversal attempt [**] [Classification: web-application-attack
06/02-15:27:23.003209   [**]  [1:460:4] ICMP Unassigned! (Type 2) [**] [Classification: misc-activity] [Priority: 3] \{ICMP\} 47.61.236.66
06/02-15:28:27.339167   [**]  [1:2153:1] WEB-PHP autohtml.php directory traversal attempt [**] [Classification: web-application-attack] [P
06/02-15:28:33.699233   [**]  [1:2399:1] WEB-PHP WAnewsletter db_type.php access [**] [Classification: web-application-activity] [Priority
06/02-15:30:25.402325   [**]  [1:893:5] WEB-CGI MachineInfo access [**] [Classification: attempted-recon] [Priority: 2] \{TCP\} 100.1.101.
06/02-15:31:19.487562   [**]  [1:2295:2] WEB-PHP Advanced Poll admin_settings.php access [**] [Classification: web-application-activity] [
06/02-15:31:46.338964   [**]  [1:1847:3] WEB-MISC webalizer access [**] [Classification: web-application-activity] [Priority: 2] \{TCP\} 1
06/02-15:32:05.098323   [**]  [1:1629:3] OTHER-IDS SecureNetPro traffic [**] [Classification: bad-unknown] [Priority: 2] \{TCP\} 189.193.2
06/02-15:32:09.072953   [**]  [1:992:5] WEB-IIS adctest.asp access [**] [Classification: web-application-activity] [Priority: 2] \{TCP\} 1
06/02-15:32:26.451484   [**]  [1:254:3] DNS SPOOF query response with TTL of 1 min. and no authority [**] [Classification: bad-unknown] [P
06/02-15:34:44.899582   [**]  [1:2082:4] RPC portmap rpc.xfsmd request TCP [**] [Classification: rpc-portmap-decode] [Priority: 2] \{TCP\}
06/02-15:35:56.989148   [**]  [1:228:1] DDOS TFN client command BE [**] [Classification: attempted-dos] [Priority: 2] \{ICMP\} 100.1.101.1
06/02-15:36:18.789820   [**]  [1:803:7] WEB-CGI HyperSeek hsx.cgi directory traversal attempt [**] [Classification: web-application-attack
06/02-15:36:54.668583   [**]  [1:1501:5] WEB-CGI a1stats a1disp3.cgi directory traversal attempt [**] [Classification: web-application-att
```

Figure 28: Alert File for IDS SnortNetwork_1

6.6 Conclusions and Future Work

The Cyber Attack Simulator presented in this paper is capable of generating IDS alert and ground truth files based on the specification of a computer network and attacks. The simulator is built with a user interface to allow the creation of various computer network configurations and attack actions. Although the current capabilities of the software are limited to producing Snort alerts, the software is set up to easily incorporate other types of IDS alerts. In order to do this, a mapping of the IDS alerts (such a Dragon, etc.) to the attack actions is needed.

There are a number of advancements and improvements that could be made to the current version of the simulation tool including the following:

- Develop a method to enable the automatic generation of multi-stage cyber attacks based on network vulnerabilities;
- Develop and integrate a probabilistic model for simulating hacker behavior including hacker capabilities based on the progress of the attacks, timing of attack actions, and deception tactics;
- Develop a commercial quality software for the Cyber Attack Simulator including mechanisms for efficiently updating the simulator to incorporate new network hardware and software, new vulnerabilities, and new hacker behaviors; and
- Investigate the extension of the Cyber Attack Simulator concept to other domains.

7 Visualization Cyber Attacks

Visualizing cyber attacks created by the INFERD engine requires a high level view and a low level view. The high level view allows for situation awareness of each attack track at the same time. The low level view presents specific information relative to each attack track and provides for impact assessment.

7.1 Related Work: VisAlert

The University of Utah has created a graphical tool, called VisAlert that periodically captures IDS alerts and associates them with the machines (where) and the time (when) the alerts happened. This tool allows visualizing the what, when, and where of IDS alerts (Figure 29).

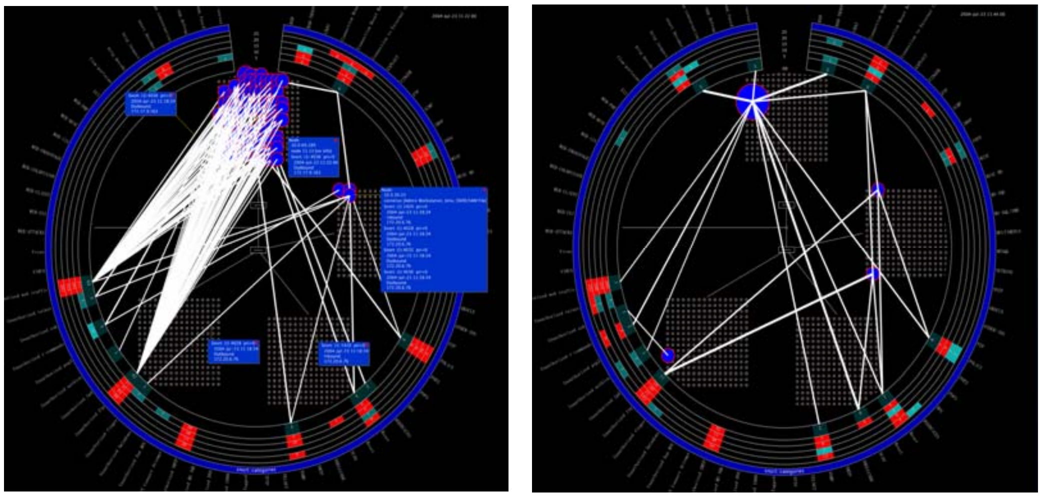


Figure 29: VisAlert (University of Utah)

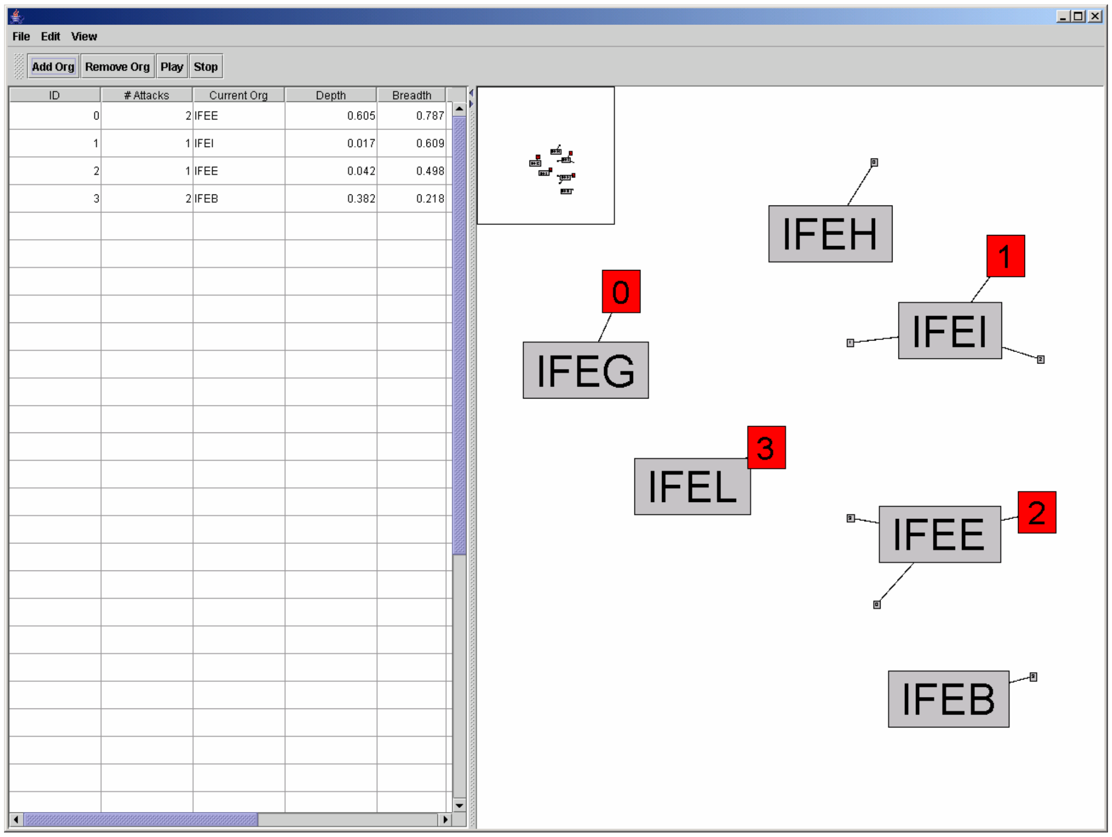
The “what” provided by the tool are the IDS alerts, they are shown on the outside of the large outer circle. The “when” is displayed via bands inside the outer circle. Each band represents a time period and contains a colored rectangle displaying the number of times a specific IDS alert occurred. The “where” is the computer network displayed in the middle. Lines are drawn from the inner most band to the computer network to give the user an idea of what alerts are occurring where.

This tool is effective at presenting how often IDS alerts occur but requires the user to have expert knowledge on alert types. It is almost impossible to monitor a complex cyber attack through pure alert level information. When alerts are numerous, the display can become congested, as shown in Figure 29. It doesn’t matter what type of information is placed on the “what” axis; if that entity becomes large, the display will cause confusion and make it difficult to provide accurate situation awareness.

7.2 Views for Visualizing Cyber Attacks

7.2.1 High Level View

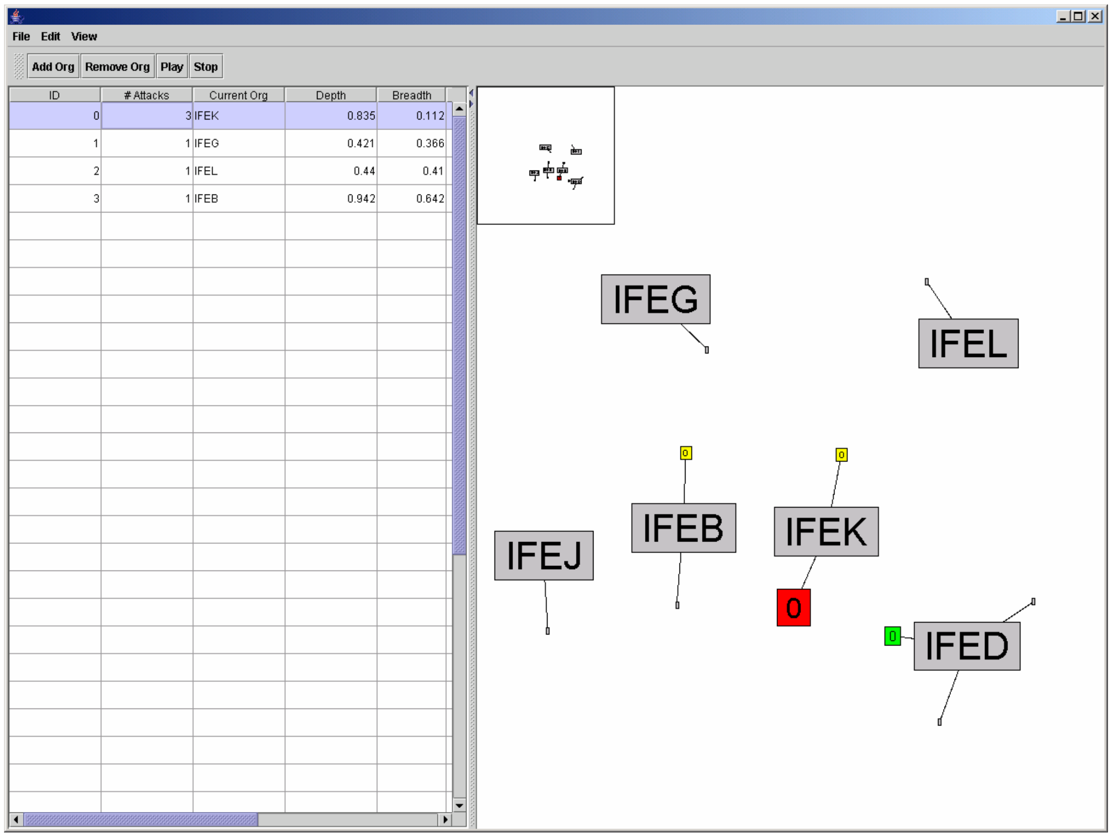
The high level is designed to allow the user to grasp how the network is performing as a whole based on the notion of INFERD attack track generation. If there are numerous attack tracks on the network, the tool presents the user with key information without cluttering the display (Figure 30). A sortable and resizable spreadsheet is used to list each attack track and their associated datum (ID, Depth Level, Breadth Level, Attack type etc). A network graph, associated with the spreadsheet, presents a graphical view of each attack track and their current situation. The graph has zooming and dragging capabilities. The large gray nodes represent organizations, which are defined by the user. Organizations can be IP classes, office locations, or any other containment that the user needs to visualize their network. Nodes extending from the organizations are the machines or subnets and are linked to the attack tracks contained in the spreadsheet. When no tracks are selected in the spreadsheet, all tracks are represented in the network graph by a red node. The red node is connected to the current organization of the track. Located in the middle of the red node is the unique ID of its associated track. The smaller nodes that extend off of the organizations represent previous or predicted attacks in the attack tracks.



ID	# Attacks	Current Org	Depth	Breadth
0	2	IFEE	0.605	0.787
1	1	IFEI	0.017	0.609
2	1	IFEE	0.042	0.498
3	2	IFEB	0.382	0.218

Figure 30: High Level Overall SituationView

If the user selects a single track or multiple tracks, the network graph changes its display to specifically represent the previous, current, and predicted situations of the selected track(s) (Figure 31). The small yellow nodes are the previous attacks and the green node is the predicted attack. Each node has the unique ID of the track in its center. When a track is removed from the spreadsheet, the nodes related to it on the network view are also removed. This prevents the display from becoming cluttered with nodes that are providing no information for the current overall situation awareness. The historical information will be stored in the database of INFERD for future analysis.



ID	# Attacks	Current Org	Depth	Breadth
0	3	IFEK	0.835	0.112
1	1	IFEG	0.421	0.366
2	1	IFEL	0.44	0.41
3	1	IFEB	0.942	0.642

Figure 31: High level view when an individual attack is selected

The high level view is effective at providing overall situation awareness because the user can assess all attack tracks simultaneously. The user knows where the current attacks are occurring (red node), what the current attacks are (spreadsheet), and when the attacks occurred (by selecting tracks in the spreadsheet).

7.2.2 Low Level View

This view provides specific information for the attack tracks. It has four associated displays. The first provides situation awareness and contains the complete history of the selected track(s) by showing how they have traversed the network in time and displays the organization, IP address, and attack type (Figure 32). Each node in the graph contains this information and the colors correlate the same to the high level view (yellow = past attacks, red = current attacks, green = predicted attacks). Up to four tracks can be compared at once.

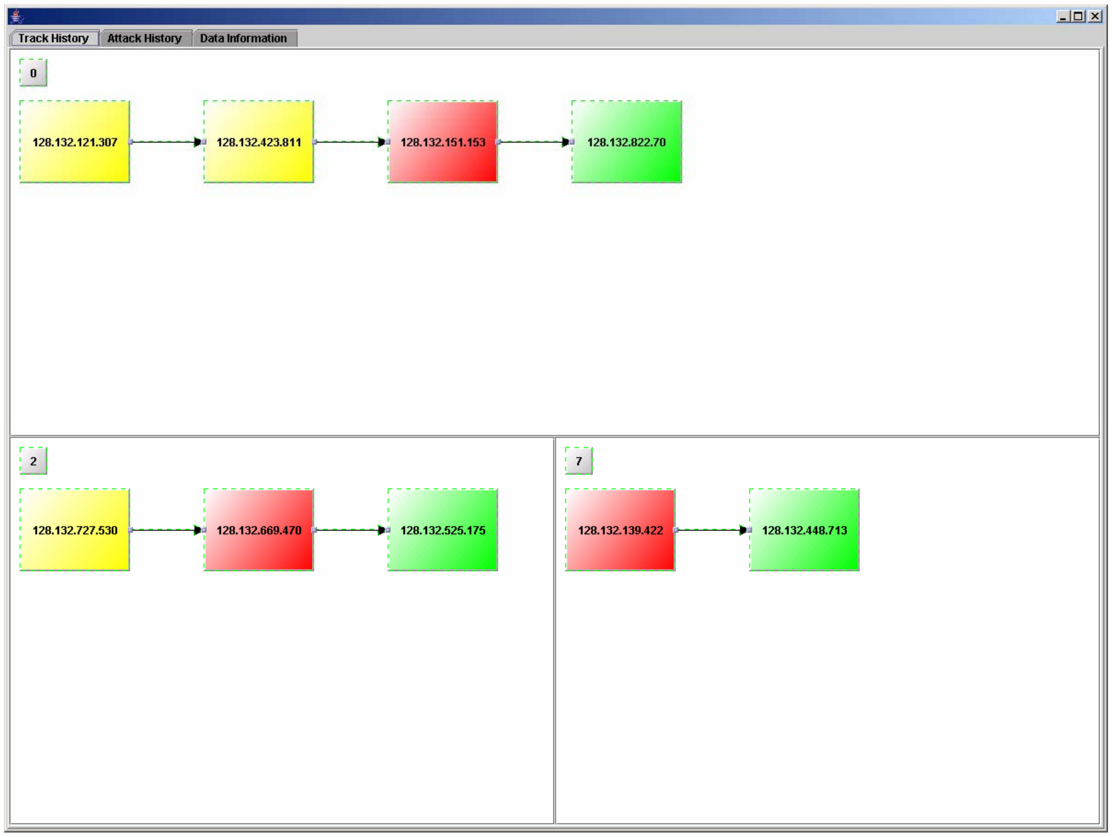


Figure 32: Track History

The second display contains the guidance template being used by INFERD (Figure 33). This display responds to selected tracks by showing how tracks have traversed the template. Because the template provides a path from beginning reconnaissance stages to ending goals, it is helpful to know the possible paths the track can take.

The third display will show impact assessment based on the work on threat assessment. It will be shown as a graph that displays which type of information could be compromised due to a specific attack. The fourth display will be a spreadsheet containing all of the alerts associated with each attack track. This will allow an interested user to drill down and get a low level idea of how INFERD is classifying tracks.

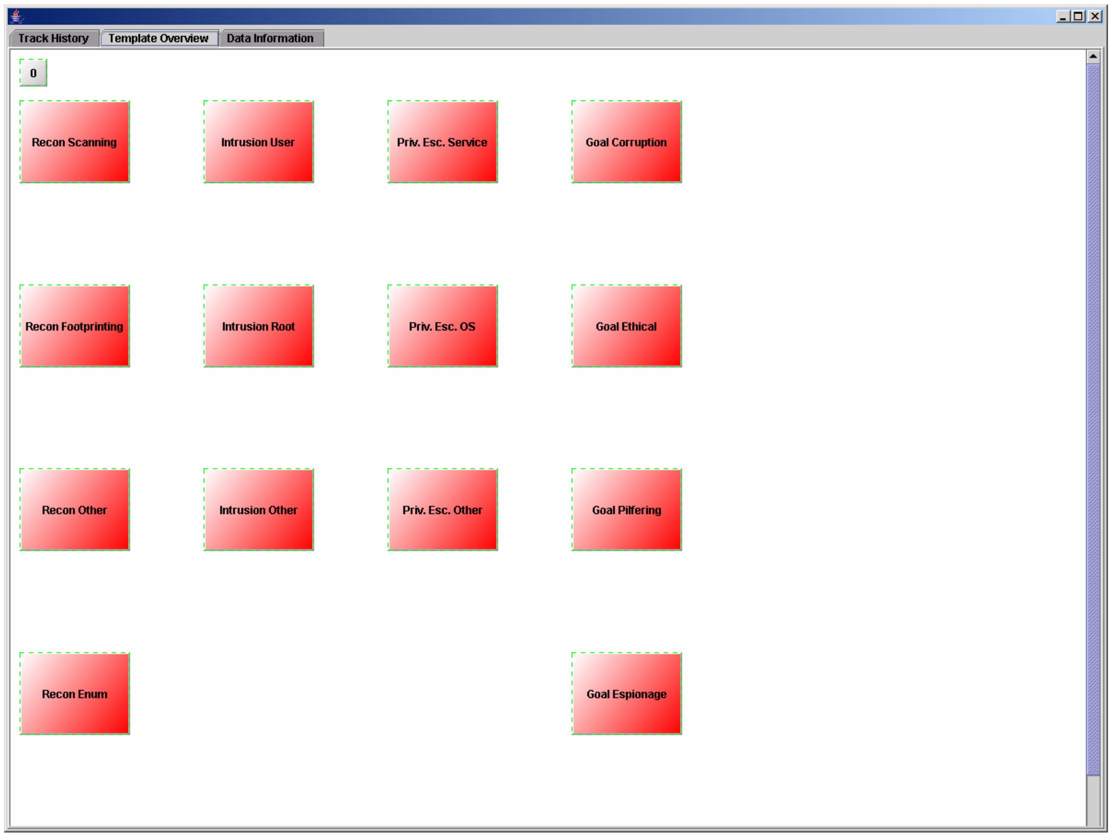


Figure 33: Guidance Template

7.3 Conclusion

The correct visualization for INFERD has to provide a means to view the entire situation at once with the ability to drill down to IDS alert level information. The two views introduced in this paper provide these functionalities. The high level view allows for situation awareness of the entire network while the low level view permits the specifics of attack tracks to be compared.

8 References

1. J. Allanach, H. Tu, S. Singh, P. Willett, and K. Pattipati, "Modeling threats," *IEEE Potentials*, vol. 23, no. 3, pp. 18–21, 2004.
2. C. Bennett and R. Jacik, "The zen of risk assessment," *Educause Quarterly*, no. 2, 2005.
3. Q. Changwen and H. You, "A method of threat assessment using multiple attribute decision making," in *Proceedings of 6th International Conference on Signal Processing*, vol. 2, August 2002, pp. 1091– 1095.
4. D. Hall, J. Llinas, Handbook of Multisensor Data Fusion, CRC Press, 2001.

5. D. L. Hall and J. Llinas, "An introduction to multisensor data fusion," in *Proceedings of the IEEE*, vol. 85, no. 1, January 1997, pp. 6–23.

6. F. Jensen, *Bayesian Networks and Decision Graphs*. Springer, 2001.
7. M. H. Kang and T. Mayfield, “A cyber-event correlation framework and metrics,” in *Proceedings of SPIE*, vol. 5107, 2003, pp. 72–82.
8. W.D. Kelton, R.P. Sadowski, and D.T. Sturrock. (2004) *Simulation with ARENA*, Third Edition, McGraw-Hill, Boston, MA.
9. J-S Lee, J-R Jung, J-S Park, and S.D. Chi. (2004) Linux-Based System Modelling for Cyber Attack Simulation. In *Proceedings of the 13th International Conference on AI, Simulation, and Planning in High Autonomy Systems*, Jeju Island.
10. E. Little and G. Rogova, “Ontology for threat assessment,” 2004.
11. D. Nicol, J. Liu, M. Liljenstam, and G. Yan. (2003) Simulation of Large-Scale Networks Using SSF. In *Proceedings of the 2003 Winter Simulation Conference*, ed. S. Chick, P. J. Sánchez, D. Ferrin, and D. J. Morrice, Institute of Electrical and Electronics Engineers, Piscataway, NJ.
12. C. Phillips and L. P. Swiler, “A graph-based system for network vulnerability analysis,” in *NSPW ’98: Proceedings of the 1998 workshop on New security paradigms*. New York, NY, USA: ACM Press, 1998, pp. 71–79.
13. J. Pierce, R. John, *An Introduction to Information Theory – Symbols, Signals, and Noise,* Dover Publishers, Inc., New York, NY, 1980.
14. M. Sudit, A. Stotz, M. Holender, *Situational Awareness of a Coordinated Cyber Attack*, SPIE Defense & Security Symposium, Orlando, FL. March 2005.
15. C. Shannon, *A Mathematical Theory of Communication*, The Bell System Technical Journal, volume 27, pages 379-423, 1948.
16. C. Taylor, A. Krings, and J. Alves-Foss, “Risk analysis and probabilistic survivability assessment (rapsa): An assessment approach for power substation hardening,” in *Proc. ACM Workshop on Scientific Aspects of Cyber Terrorism*, Washington, D.C., November 2002.
17. C. Tsallis, *Nonextensive Statistics: Theoretical, Experimental and Computational Evidences and Connections*, Brazilian Journal of Physics, Vol. 29, No. 1, March 1999.
18. C. Tsallis, *Entropic Nonextensivity: A Possible Measure of Complexity*, Chaos Solutions and Fractals, Vol. 13, 371-391, 2002.
19. S. Vidalis and A. Jones, “Using vulnerability trees for decision making in threat assessment,” University of Glamorgan, School of Computing, Tech. Rep. CS-03-2, June 2003.
20. R. R Yager,. *On Ordered Weighted Averaging Aggregation Operators in Multi-criteria Decision Making*, IEEE Transactions on Systems, Man and Cybernetics 18, 183-190, 1988.
21. R. R. Yager, *Hierarchical Aggregation Functions Generated from Belief Structures*, IEEE Transactions on Fuzzy Systems, Vol. 8, No. 5, 481-490, October 2000.
22. R. R. Yager, *Generalized OWA Aggregation Operators*, Fuzzy Optimization and Decision Making, 2, 93-107, 2004.
23. N. Ye, Y. Zhang, and C. M. Borror, “Robustness of the markov-chain model for cyber-attack detection,” in *IEEE Transactions on Reliability*, vol. 53, no. 1, Mar. 2004, pp. 116–123.
24. ——, “Detecting, tracking and counteracting terrorist networks via hidden markov models,” in *IEEE Aerospace Conference Proceedings*, Mar. 2004, pp. 3246–3257.

APPENDIX A: Cyber Attack Simulator (Version 1.0): User's Guide

The Cyber Attack Simulator is designed to model cyber attacks in a computer network for the purposes of generating alerts produced by host-based and network intrusion detection systems such as Snort. The simulator is written using a commercially available simulation software called ARENA (version 7.0) produced by Rockwell Software (http://www.arenasimulation.com). This software must be installed on the computer to use the simulator. This User's Guide is primarily focused on the use of the Cyber Attack Simulator. For additional information on the use of ARENA, please refer to the ARENA help menu or [8].

Software Setup

Before you begin, you will need to download the files described in Table A1 to a new folder on your computer. A CD containing the Cyber Attack Simulator files has been submitted to the Air Force Research Lab - IFEA

Table 2: Files Required to Use the Cyber Attack Simulator

File Name	Description
Cyber Attack Simulator v1.doe	Cyber Attack Simulator Template File
ListofActions.txt	Attack Actions User Can Select From
SnortAlertDefs.txt	Definition of Snort Alerts Mapped to Attack Actions
SnortPriorityDefs.txt	Defined Priorities of Snort Alerts
HttpInspectDefs.txt	Definition of Classified Snort http Inspect Messages
DragonAlertDefs.txt	Definition of Dragon Alerts Mapped to Attack Actions
Network.tpo	ARENA Module Template File

To use the software, begin by opening the template file "Cyber Attack Simulator v1.doe" using the ARENA simulation software. Upon opening, the ARENA window should display, a simulation logic diagram like the diagram displayed in Figure 34. **<u>DO NOT EDIT THIS LOGIC</u>**. This model logic is required for the Cyber Attack Simulator to function properly.

On the left side of the ARENA interface are template panels that contain modules represented by icons. If the "Network" template is not already attached, using the mouse, right-click in the template area and select attach. Then browse to the location of the file "Network.tpo" and attach it.

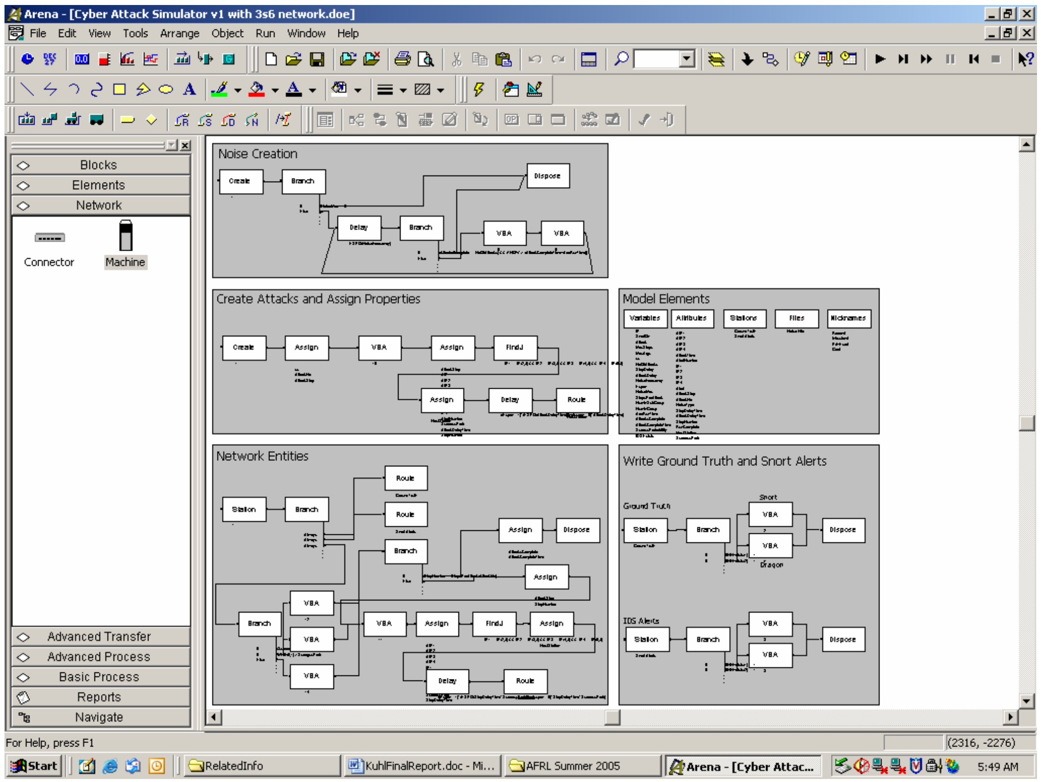


Figure 34: Simulation Logic for the Cyber Attack Simulator

Constructing the Computer Network

To begin constructing a computer network, scroll to a blank area of the modeling window. Using the mouse, left click on the "Machine" or "Connector" icons in the "Network" template. Drag and drop the icon onto the modeling window. Once the machines and connectors have been added, the connector icon on the ARENA tool bar, , can be used to connect make connections among the machines and connectors. Figure 35 displays an example of a subnet consisting of a connector and two machines.

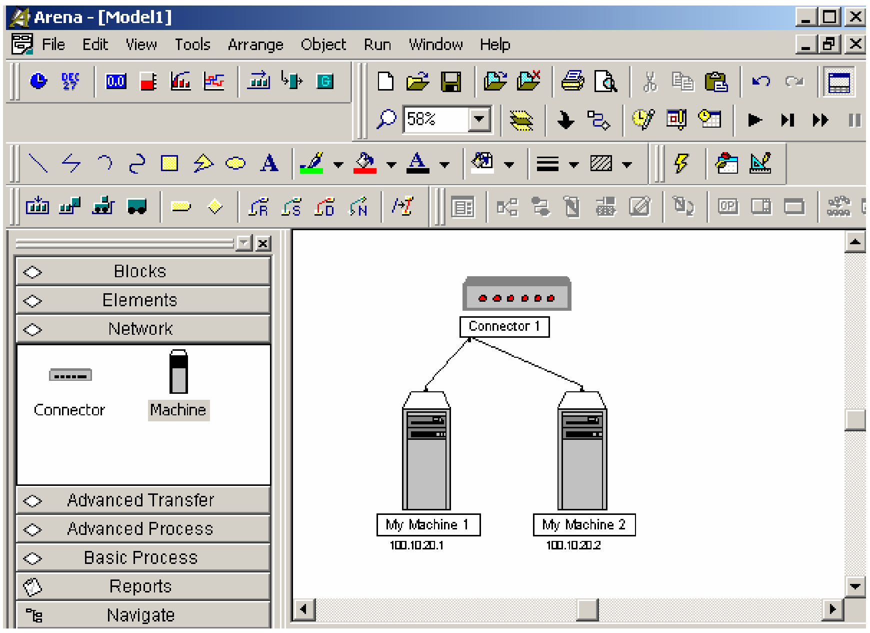
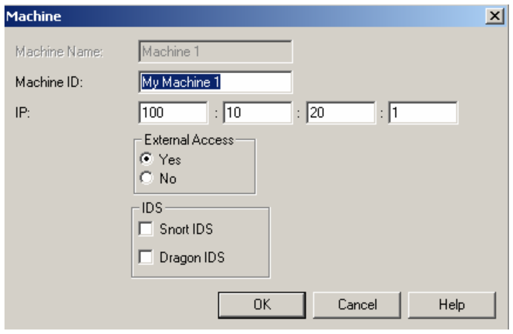


Figure 35: Network Example

To specify the detailed information about the machines, double-click on the box labeled "My Machine #". The "Machine" dialog box (Figure 36) will appear with the following input:

- The "Machine Name" is used internally in the software an cannot be changed by the user.
- The "Machine ID" is a user specified, descriptive name for the machine. A default machine ID is generated.
- The "IP" is a user specified IP address for the network. A default IP address is generated.
- "External Access" - "Yes" specifies that the machine can be accessed from outside the network (e.g. the Internet); "No" specifies that the machine can only be accessed by machines within the network (further access limitations may be stipulated based on the setup of the subnets and connectors.
- "IDS" – Check the boxes for the host-based IDSs that should be operational during the simulation on this machine. A separate report file will be created for each IDS specified.

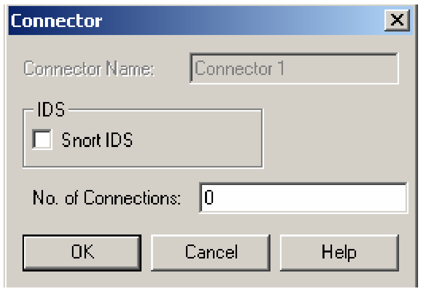

Machine

Machine Name: Machine 1
Machine ID: My Machine 1
IP: 100 . 10 . 20 . 1
External Access: [✓ Yes] No
IDS: ☐ Snort IDS ☐ Dragon IDS

Figure 36: Machine Dialog Box

To specify the detailed information about the connectors, double-click on the box labeled "Connector #". The "Connector" dialog box (Figure 37) will appear with the following input:

- The "Connector Name" is used internally in the software an cannot be changed by the user.
- "IDS" – Check the box for the network IDS that should be operational during the simulation on this connector. A separate report file will be created for each IDS specified.
- "No. of Connections" is used to specify the number of connectors/subnets that this connector/subnet connects to.


Connector

Connector Name: Connector 1

IDS

☐ Snort IDS

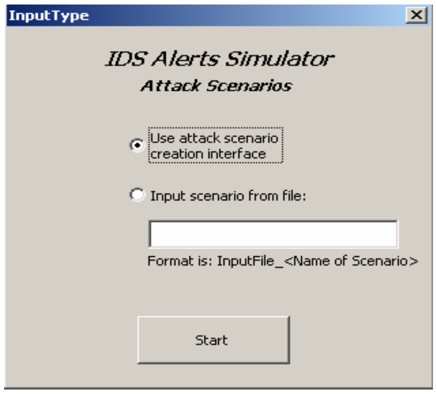
No. of Connections: 0

Figure 37: Connector Dialog Box

Specifying Attacks and Running the Cyber Attack Simulator

Before proceeding to the attack specifications, the user can update the starting time for the attacks by selecting RUN → SETUP from the ARENA menu bar and changing the "Start Date and Time".

To specify the attacks and run the Cyber Attack Simulator, click on the "Go" button in the ARENA toolbar. The "Input Type" specification dialog box (Figure 38) will appear. The user can specify to use the attack specification interface to create an attack scenario or if a scenario has already been defined previously, the user can specify the file name for the scenario. Upon creating a new scenario, an input file is automatically created that saves all of the user input with regard to the attack scenario. The file name has the form "InputFile_<Name of Scenario>".


InputType

IDS Alerts Simulator

Attack Scenarios

● Use attack scenario creation interface

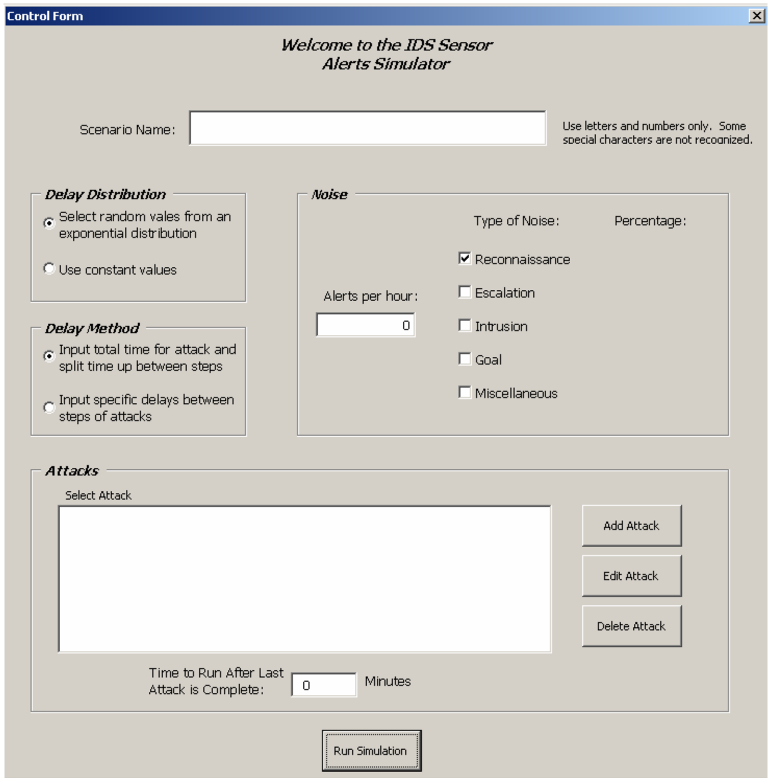
○ Input scenario from file:

Format is: InputFile_<Name of Scenario>

Figure 38: Input Type Specification Dialog Box

If “User Attack Scenario Creation Interface” is selected, the scenario specification dialog box (Figure 39) will appear. In this form you can specify the following aspects of the attack scenario:

- “Scenario Name” - Enter a name for the scenario, this name will be used in naming the files that are created when the simulator is run.
- “Delay Distribution” – Select whether the delay times between steps in the attack should be user specified constants or if the delay time should be sampled from an exponential distribution with a user specified mean value.
- “Delay Method” – Select whether the user will specify the average total time for the attack or if the user would like to specify the average delay time between each attack step.
- “Noise” – Specify the rate that noise alerts should generated during the simulation. By default, 100% of the alerts will be generated from the Reconnaissance category of attack actions. The user can specify by checking the box associated with the categories of noise alerts to generate. Then the user can specify the percentage of alerts of each category. These noise alerts will be randomly generated.
- “Attacks” - The user can add, edit, or delete any of the attacks in the list. If “Add Attack” is selected the attack specification dialog box (Figure A7) appears (see below). Also the user can specify the average length of time the simulation should run after the last attack has ended.


Control Form

Welcome to the IDS Sensor Alerts Simulator

Scenario Name:

Use letters and numbers only. Some special characters are not recognized.

Delay Distribution
- [✓] Select random vales from an exponential distribution
- [ ] Use constant values

Delay Method
- [✓] Input total time for attack and split time up between steps
- [ ] Input specific delays between steps of attacks

Noise

Alerts per hour: 0

Type of Noise: Percentage:
- [✓] Reconnaissance
- [ ] Escalation
- [ ] Intrusion
- [ ] Goal
- [ ] Miscellaneous

Attacks

Select Attack

Time to Run After Last Attack is Complete: 0 Minutes

Figure 39: Scenario Specification Dialog Box

To specify the details of the attack, when the “Attack Specifications” dialog box (Figure 40) appears, the following information can be specified:

- “Attack Name” – Enter a descriptive name for the attack
- “How many steps would you like for this attack?” – Enter the number of steps in the attack and click “Reset Form”. The form will display a multiform with one tab for each attack step.
- For each step, the user can specify the following information for the attack:
    - “Source IP” – Select the attacking machine IP address or External from the drop-down menu. If “External” a random IP address for a computer outside the network will be generated.
    - “Target IP” – Select an IP for the target machine for this step of the attack from the drop-down menu. This list will be limited to the computer with which the “Source IP” can communicate.
    - “Probability of Success” - Specify the probability that this attack step will be successful. If a value strictly between 0 and 1 is specified, the success of the step will be determined randomly and the step will be repeated until it is successful. If 0 or 1 is specified, the step will be executed only once.
    - “Encode This Step?” – Specify whether the action for this step has been encoded by the user.
    - “Action” – Select a group from which an action should be selected. Then click “Update Subgroups”. At this point, the user can specify a subgroup. If no subgroup is selected, the action will be generated randomly from the group specified. If a subgroup is selected, click “Update Actions”. At this point, the user can specify an action for this step. If no action is selected, the action will be generated randomly for the subgroup specied.
    - If the user specified, “Input Specific Delays Between Steps of the Attacks” in the scenario specification dialog box, a box will appear on the tab for each attack step where the user can specify the average time until the next step of the attack. If not, a box specifying the average total time for the attack will appear on the tab for Step 1 of the attack.
- “Delay This Attack By” – Specify the average amount of time the start of the attack should be delayed from the beginning of the simulation.

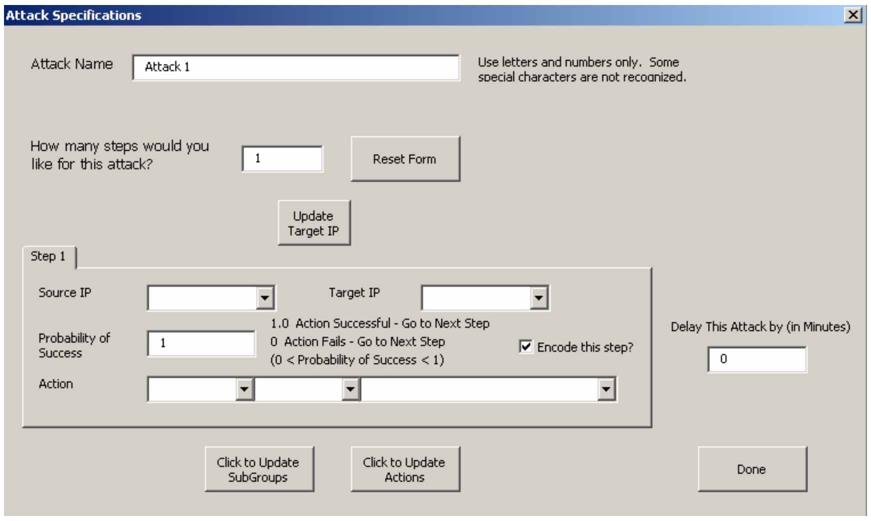

Attack Specifications
Attack Name
Attack 1
Use letters and numbers only. Some special characters are not recognized.
How many steps would you like for this attack?
1
Reset Form
Update Target IP
Step 1
Source IP
Target IP
Probability of Success
1
1.0 Action Successful - Go to Next Step
0 Action Fails - Go to Next Step
(0 < Probability of Success < 1)
Encode this step?
Delay This Attack by (in Minutes)
0
Action
Click to Update SubGroups
Click to Update Actions
Done

Figure 40: Attack Specification Dialog Box

Simulation Output

After entering the all of the information about the attack scenario, click “Run Simulation” on the Scenario Specification dialog box. When the simulation is complete, a dialog box (Figure 41) will appear listing the simulation output files. From this dialog box, the output files can be viewed, the file names can be edited, or the files can be deleted. Unless the files are deleted, they will be saved in a folder with the name of the scenario.

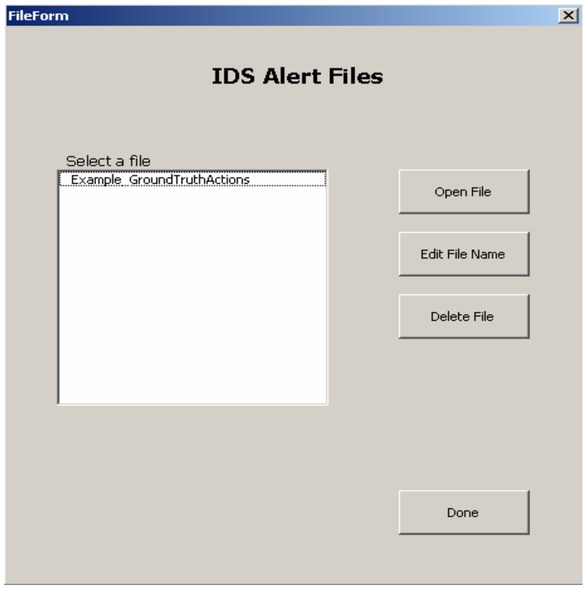


Figure 41: Output File Dialog Box